Universidad Politécnica de Madrid
Departamento de Inteligencia Artificial

Máster Universitario en Inteligencia Artificial

A Theoretical Consideration of the parameters of the Max Min Ant System

Supervisor: Nik Swoboda

Mariano Gentile

Date: 2015 - 12 - 01

Abstract

In this thesis we will discuss the setting of the parameters of the Max-Min Ant System. In the literature it is possible to find theoretical and practical considerations of these parameters, nevertheless it seems that they have not been studied in a joint manner. We propose a theoretical study of the relationship between them, giving the user some further knowledge at the time of setting the algorithm's parameters and some new idea are proposed. In particular, the number of ants is studied in more detail. Then we will study the settings of the τ_{max} and τ_{min} in a way which is different from the most commonly used technique, taking in consideration theoretical as well as experimental problems. Finally, some experiments are shown that demonstrate the validity of our proposals.

Contents

Chapter 1

Introduction

A well known problem in Computational Complexity Theory is to find an optimal polynomial solution for a NP-complete problem. NP problems, which acronym means "non-deterministic polynomial time" (and NP-complete problems too, as they are a subset of NP) are characterized by the computational time needed to find a solution that behaves super-polynomially with the size of the problem itself. Nowadays a polynomial algorithm is not known to solve any of these problems. The importance of solving a NP-complete problem depends on the fact that any other NP problem can be polynomially reducible to any NP-complete problem. This means that a polynomial algorithm built for one of them can be polynomially transformed into an algorithm for all the known NP problems.

Recent attention has settled on The Travelling Salesman Problem (TSP) since it is a NP-complete problem. As its name suggests, it can be viewed as the problem of a salesman that must visit a set of cities (or a set of customers), using the shortest route.

As already mentioned, a polynomial algorithm to find an optimal solution to the TSP is not known. Many theoretical and experimental studies focus on the possibility of finding a "good" solution in a prefixed computational time. These algorithms have a stochastic component that needs to be carefully studied. This is indeed the case of the Ant System.

In 1989 some studies of ant species behaviour in Argentina, as for example [13], showed a technique used by these little insects that is capable of finding a good path to the nearest food source (in other words, they are capable of solving the "Shortest Path Problem"). This is surprising considering that they have no visual or acoustic sense, unlike humans and most of the higher species on Earth. These studies concluded that ants can reach their objective using only rudimentary communication between them: each ant leaves a chemical substance produced by itself on the ground when it is moving, called "pheromone", so that the next ant is tempted to follow this trail. In a colony of ants, each ant tends to follow the path with the biggest amount of pheromone, creating a stochastic behaviour that ends with finding a good path to the food source. Over time more and more ants follow this path leaving more pheromone, while the pheromone on the other paths evaporates, thus allowing most of the colony to follow only one path. References [13, 11] describe in more detail this behaviour and the problems of the method used by real ants.

The Ant System's idea arises from the observation of ants' behaviour but with some differences from the natural system, as explained in [11], due to some practical aspects developed with the goal of improving the solution.

Chapter 2

Background and Previous Work

2.1 Travelling Salesman Problem (TSP)

The TSP problem can be seen as the problem of a salesman who needs to visit n cities using the shortest possible path. In mathematical terms, we can formalize the problem as follows:

Definition 1. *Let $G = (V, E, w)$ be a weighted undirected complete graph where:*

- *V is the set of nodes and $card(V) = n$ is the number of nodes,*
- *E is the set of arcs and $card(E) = n(n-1)/2$ is the number of arcs,*
- *$w, card(w) = n(n-1)/2$ are the weights on these arcs.*

Let $\mathcal{H}_G$ be the set of Hamiltonian cycles on G and $f : \mathcal{H}_G \to \mathbb{R}$ a function from the Hamiltonian cycles to real numbers, called the fitness function.

The Travelling Salesman Problem is the problem of maximizing (or minimizing depending on what the fitness function represents) f:

$$max(f(h)) \quad : \quad h \in \mathcal{H}_G$$

.

A lot of variants of the TSP problem can be found in the literature. The most popular ones are:

- symmetric/asymmetric TSP: this difference involves the set w, in particular they differ in whether the cost of going from any city A to any city B is equal to going from the city B to the city A or not.
- Euclidean TSP: we consider an Euclidean space where the nodes of the graph lie. Suppose there are no obstacles between them, so that it is possible to follow a straight line to reach any city B from any city A. With these conditions the weights between each pair of nodes are the Euclidean lengths of the respective arcs and the fitness function is the length of the Hamiltonian path constructed, that is $f(h) = length(h)$ (in this case we are considering a minimization problem).
- TSP with time windows: in this variant the customers have a period of time in which they can receive the salesman. If the salesman goes to any city A out of its time window, the fitness function decreases.
- Probabilistic TSP: when reaching a city, there is a probability that the customer will not receive the salesman; the probability function can vary with the city and some other variables.

These are just some examples of TSP variants, and it is also possible to mix them in order to find others even more complex. In this work we will analyse the simplest one, i.e. the Symmetric Euclidean TSP Problem, formally:

Definition 2. *Symmetric Euclidean TSP Problem*
Let $G_{SE} = (V, E, w)$, as in Definition 1, with the following restrictions:

- *w are symmetric, that is*
  $w(E(i,j)) = w(E(j,i))$, $\forall i, j = 1, \ldots, card(V)$ (in practical situations we will consider the case $i \neq j$),
- *the length of an arc is the Euclidean distance between its nodes, that is*
  $w(E(i,j)) = dist(i,j)$, $\forall i, j = 1, \ldots, card(V)$ (recall that usually $i \neq j$).

Let $\mathcal{H}_{G_{SE}}$ be the set of Hamiltonian cycles on G_{SE} and $f : \mathcal{H}_{G_{SE}} \to \mathbb{R}$: $f(h) = length(h)$ the function from the Hamiltonian cycles to the real numbers, that associates each Hamiltonian cycle with its length.
The Travelling Salesman Problem is the problem of minimizing f:

$$min(f(h)) \quad : \quad h \in \mathcal{H}_{G_{SE}}$$

Note that the size of the solution set for the TSP problem is equal to $n!$ where n is the number of cities. This is the basic intuition as to why the TSP problem is in NP.

2.2 Ant Colony Optimization (ACO)

Studies of ants realized in the late 80s, started with the idea of making virtual agents communicate through the environment, as real ants do. These virtual agents leave "messages" in the environment, that other agents can intercept and analyse. Note that there is a difference with other algorithms, namely where the information is located: in genetic algorithms for example each agent holds its own information, changing and elaborating

the information inside themselves. In Ant Systems algorithms the information is left in the environment and the process that modifies it is external to the agents.

The first attempts to use these algorithms tried to solve problems related with graph problems, as for example the Shortest Path Problem (the one real ants try to solve), or the Travelling Salesman Problem. In this kind of problem the environment is a graph (normally a complete graph) where virtual ants move from one node to another. At each iteration of the algorithm, each ant builds a solution based on a probability function that depends on the pheromone and on some heuristics that the user can add, and uses it to choose the next node it will visit. At the end of this phase, the fitness of all solutions is calculated, and the algorithm leaves some pheromone on the arcs of the best one, based on the quality of the found solution. In order to reproduce the real world behaviour of ants, the virtual pheromone on arcs evaporates, allowing bad solutions to become (almost) inaccessible in the future. The main references where we can find the beginnings of Ant System ideas are [9, 10, 8, 11]. To implement the algorithm we have described we need to define some parameters:

- $end\_condition$: the condition set to stop the algorithm
- $number\_of\_ants$: the number of ants used
- α: a parameter in $[0, \infty]$ that regulates the importance of the pheromone
- β: a parameter in $[0, \infty]$ that regulates the importance of the heuristic function
- ρ: a parameter in $[0, 1]$ that regulates the evaporation rate of the pheromone
- $\Delta\tau_{i,j}$: the function that regulates the amount of pheromone added to the arcs of the best solution

- τ_0: the initial pheromone

These parameters strongly change the final result of the algorithm, and they have been thoroughly studied in literature. We will briefly analyse the main results of them separately. Then, we will discuss them in a joint manner.

A pseudo-code of a basic implementation of the algorithm is as follows:

1. initialize the pheromone matrix: $\forall i, j = 1, \ldots, n : \tau_{i,j} = \tau_0$
2. initialize the heuristic function: $\forall i, j = 1, \ldots, n : \eta_{i,j} = g(E(i, j))$ where $g : E \to \mathbb{R}$ is a function chosen by the user and E the set of arcs in the graph (usually the case $i = j$ is excluded)
3. initialize the global best solution: $l_{gb} = +\infty$, $t_{gb} = null$ where l_{gb} is the length of the solution and t_{gb} the array containing the arcs in the solution
4. *while* (*not end_condition*):
5. initialize the iteration's best solution index: $i_b := null$
6. initialize the iteration's best length: $l_{i_b} := null$
7. initialize the iteration's best tour: $t_{i_b} := null$
8. *for* $k := 1, \ldots, number\_of\_ants$:
9. initialize the tour of ant k: $t_k := null$
10. initialize the length of the tour of ant k: $l_k := \infty$
11. initialize the cities visited set: $C := null$

12. select the first city $i \in V$, where V is the set of nodes of the graph and *add* i *to* C

13. *while* ($\exists$ *city* $\notin C$):

14. choose the next city $j \in V,\ j \notin C$ to be visited with probability

$$p_j := \frac{\tau_{i,j}^{\alpha} \eta_{i,j}^{\beta}}{\sum_{j \notin C} \tau_{i,j}^{\alpha} \eta_{i,j}^{\beta}}$$

15. *append* (i, j) *to* t_k and *add* j *to* C

16. change the current city: $i := j$

17. *endwhile*

18. $l_k := length(t_k)$

19. *endfor*

20. check the iteration's best solution index: $i_b := argmin_k(l_k)$

21. set the iteration's best length: $l_{i_b} := l_k \quad : \quad k = i_b$

22. set the iteration's best tour: $t_{i_b} := t_k \quad : \quad k = i_b$

23. $if\ l_{i_b} < l_{gb} \Rightarrow (\ l_{gb} := l_{i_b}$ and $t_{gb} := t_{i_b}\ )$

24. make the pheromone evaporate: $\forall i, j : \tau_{i,j} := (1 - \rho)\tau_{i,j}$

25. increase the pheromone on the arcs of the best solution: $\forall (i, j) : \tau_{i,j} := \tau_{i,j} + \Delta\tau_{i,j}$

26. *endwhile*

27. return the global best solution found: l_{gb}, t_{gb}

As already mentioned, the first implementations of Ant Systems were based on graph related problems. Recently, many researcher have focused their attention on what it is called a *meta-heuristic*. The concept of meta-heuristic can be described as a set of algorithms based on heuristic methods, that are applicable to a wide set of different problems. Following this idea, the Ant System was generalized into the *Ant Colony Optimization Meta-Heuristic*, or *ACO*. A detailed description of ACO can be found in [25, 8, 11]. Here we will mention its main definition:

Definition 3. *The ACO encoding*

- *let* $(\mathcal{S}, f, \Omega)$ *be a combinatorial maximization (or minimization) problem where* $\mathcal{S}$ *is the set of possible solutions,* $f(s)$ *the fitness function that we want to maximize (or minimize) and that depends on the solution, and* Ω *a set of constraints*
- *the goal of the algorithm is to find a global best solution*
- *map* $(\mathcal{S}, f, \Omega)$ *onto a problem characterized by:*
  - *a finite set of components* $C = c_1, \ldots, c_{N_C}$ *where* N_C *is the number of components*
  - *the possible states of the problem are defined by sequences* $x = (c_i, c_j, \ldots, c_h, \ldots)$ *of finite length, where* $c_i \in C$
  - *let* $\mathcal{X}$ *be the set of all possible states and* $\mathcal{S}$ *the subset of admissible solutions, that is* $\mathcal{S} \subseteq \mathcal{X}$
  - *let* $\overline{\mathcal{X}}$ *be the set of feasible states; for any feasible state* $x \in \overline{\mathcal{X}}$ *it is not impossible to complete it and thereby make a solution* $s \in \mathcal{S}$ *satisfying the constraints* Ω *(note that this definition does not guarantee the existence of such a solution)*
  - *a non-empty set of optimal solutions* $\widetilde{\mathcal{S}} \subseteq \overline{\mathcal{X}}$, $\widetilde{\mathcal{S}} \subseteq \mathcal{S}$

- *let $G_C = (C, L)$ the complete graph whose nodes are the components of C, called the construction graph*
- *let artificial ants build solutions over C with a randomized walk*
- *implement the constraints Ω over the set C in a hard way (ants can build only feasible solutions) or in a soft way (ants can construct infeasible solutions that will be strongly penalized by the fitness function)*

The ACO framework can solve a large set of combinatorial optimization problems, as for example the Quadratic Assignment Problem (QAP). Usually, the main problem of this implementation is to map $(\mathcal{S}, f, \Omega)$ into a problem of the described characteristics. Figure 2.1 shows a trivial implementation of a maximizing problem such as the QAP. Making use of the ACO encoding, it tries to solve the Shortest Path Problem from node 0 to node 12, where obviously the fitness function is different from the Euclidean distance.

2.2.1 The end condition

The end condition defines the instant when we will stop the algorithm and the global solution found at that point will then be considered the final one. The choice of this condition is completely free, depending on the context of the problem that we are using the algorithm for. Some possible conditions are:

- the total time
- the number of iterations
- the number of tours built (the number of tours is equal to the number of ants multiplied by the iterations)

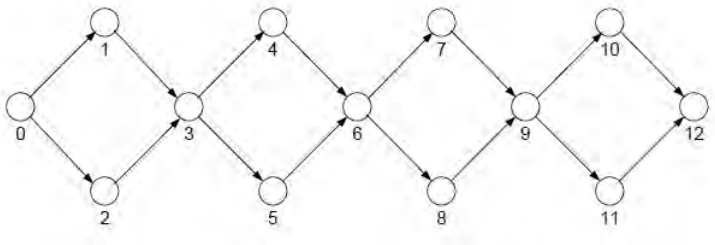

Figure 2.1: *A trivial example where the domain of a function* $f : \{0,1\}^4 \to \mathbb{R}$ *is converted into a graph. The four variables of the function are related respectively with the couples of nodes* $\{1,2\}$, $\{4,5\}$, $\{7,8\}$ *and* $\{10,11\}$. *The nodes* $1,4,7,10$ *allow their respective variables to take the value 1 and the nodes* $2,5,8,11$ *allow them to take the value 0. The nodes* $0,3,6,9,12$ *are added in order to create a graph compatible with the Shortest Path Problem, but they are not involved in the function* f. *Source: [14]*

- a predetermined solution is found
- the improvement of the global solutions found in the last k iterations is less than an ϵ chosen by the user

In our experiments we use a fixed total time. Note that the time and the number of constructed tours are not proportional in the algorithm used (the MaxMin Ant System, which will be described in Section 2.4), due to the background processes that organize the pheromone matrix.

2.2.2 The number of ants

The number of ants used in the implementation of the algorithm can change its effectiveness. Experimental discussions can be found in [11] and in many other articles, while a short theoretical study that confirms their conclusions are in [27].

We can summarize all the observations by saying that at each iteration of the algorithm, each ant uses the accumulated knowledge inside the pheromone matrix to construct a solution. So at each iteration we have m built tours with the accumulated knowledge till that moment, where m is the number of ants. The higher the number of ants, the greater the exploration at each iteration. On the other hand, the higher the number of iterations, the higher the accumulated knowledge, and the higher the exploitation. So, there is a trade-off between exploration and exploitation considering a limited runtime, as shown in the formula $iterations \approx tours/m$, where the number of tours is limited by the maximum runtime set.

Moreover, it is important to consider the size of the problem: bigger problems usually need a bigger number of ants, otherwise the exploration done at each iteration becomes too small.

As a rule of thumb, a number of ants set to 25 seems to be commonly accepted. Nevertheless, we insist on the fact that it can change considerably with the instance of the considered problem and the available running time.

In the results section we confirm the considerations given above with examples and also with a theoretical discussion that shows a different perspective of this problem.

2.2.3 The α parameter

The α parameter regulates the importance of the pheromone knowledge. The value of this parameter commonly used in the literature is 1, coinciding with the value some species of ants have. A higher value increases the importance of the pheromone values on arcs, while a lower value decreases it. Increasing α makes the algorithm exploit the knowledge accumulated in the first iterations. Note that in the starting iterations, all the arcs have the same pheromone value, so the solutions found first are completely random, or in some way are managed by the heuristic function, if it is implemented.

This means that a higher value of α increases the probability of falling into a local optimum. An interesting theoretical study about this parameter can be found in [24]. We repeat here its main arguments.

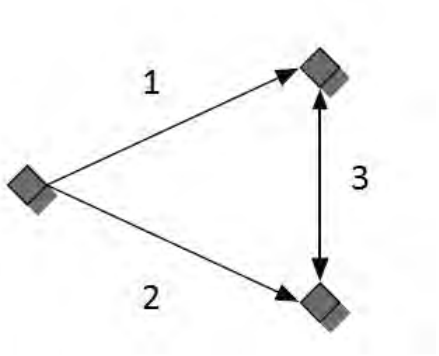

Figure 2.2: *Toy model sketch with three nodes. Source: [24]*

Let us study the simple graph in Figure 2.2, composed by 3 nodes, and suppose that there is no heuristic function ($\beta = 0$). We always start from the left node, since starting from the same node at each iteration is a common practice. We have two possibilities, go up (arc 1) or go down (arc 2), while the choice of the final arc is forced. It is important to notice that we can find this reduced model in every big problem, every time an ant needs to decide between two arcs. We can assume the right arc (arc 3, that is bidirectional) in the figure is the tour that include all other nodes not yet considered, or just the final arc if we consider the little model be the last three nodes considered at each iteration. Let us call l_1, l_2 the lengths of arcs 1 and 2. Remembering the complete formula to update pheromone $\forall i = 1, 2 \ : \ \tau_i := (1 - \rho)\tau_i + 1/l_i^{-1}$ if l_i is chosen, we want to study the steady state, when the pheromone update is zero. It implies the condition

$$\forall i = 1, 2 \ : \ \rho \cdot \tau_i \ = \ \frac{1}{l_i} \cdot m \cdot p_i^*$$

where m is the number of ants and p_i^* is the steady state of the probability over arcs 1 and 2. It is important to notice that this reasoning makes sense if we consider that all ants leave pheromone at each iteration or equivalently if we use just one ant. From the previous condition, and remembering that $\forall i = 1, 2 : p_i = \tau_i^\alpha / (\tau_1^\alpha + \tau_2^\alpha)$ the following calculations can be done, giving as a result a relation among the steady state of the probability, the α parameters and the lengths l_1, l_2:

$$\frac{p_1^*}{p_2^*} = \frac{\tau_1 \cdot l_1}{\tau_2 \cdot l_2}; \quad \frac{\tau_1^\alpha}{\tau_2^\alpha} = \frac{\tau_1 \cdot l_1}{\tau_2 \cdot l_2}; \quad \frac{\tau_1^{\alpha-1}}{\tau_2^{\alpha-1}} = \frac{l_1}{l_2}; \quad \frac{\tau_1^{1-\alpha}}{\tau_2^{1-\alpha}} = \frac{l_2}{l_1}; \quad \frac{\tau_2}{\tau_1} = \left(\frac{l_1}{l_2}\right)^{\frac{1}{1-\alpha}}$$

Substituting this relation in the probability formula:

$$p_1 = \frac{\tau_1^\alpha}{\tau_1^\alpha + \tau_2^\alpha} = \left(\frac{\tau_1^\alpha + \tau_2^\alpha}{\tau_1^\alpha}\right)^{-1} = \left(1 + \frac{\tau_2^\alpha}{\tau_1^\alpha}\right)^{-1} = \left(1 + \left(\frac{l_1}{l_2}\right)^{\frac{\alpha}{1-\alpha}}\right)^{-1} \tag{2.1}$$

From equation 2.1 we can notice that the probability of choosing arc 1 depends on the α parameter as well as the ratio between the lengths of the arcs. Note that the value $\alpha = 1$, the one commonly used by researchers, is a singularity.

In Figure 2.3 the stability analysis of equation 2.1 is shown, considering $l_1 < l_2$. We can notice that when $\alpha < 1$ we have two unstable trivial fix-points at $p_1 = 0$ and $p_1 = 1$ and one stable fix-point p_1^* while when $\alpha = 1$ we have just one stable fix-point where all ants end up following the best path. When $\alpha > 1$, the two trivial fix-points become stable while we have a third unstable fix-point. In this situation the final solution depends on the initial condition, determined by the first iterations of the algorithm. Hence there is a real risk to end up with the worst path. Considering also that the scope of the Ant System algorithm is to find the best tour just once, it is not necessary that all ants follow the best path at every iteration. For small problems, choosing $\alpha < 1$ can promote diversity. Nevertheless, this can cause the algorithm to take a long time to converge. Thus, the

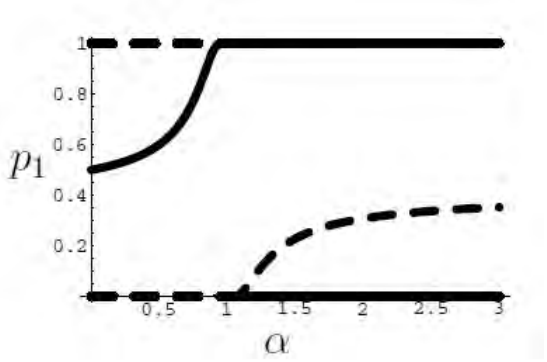

Figure 2.3: *Stability Analysis. The probability of choosing arc 1 (p_1) is represented on the y-axis, while α on the x-axis. The continuous lines represent stable fixed-points, while the dashed lines represent unstable ones. Source: [24]*

choice of $\alpha = 1$ seems to be theoretically proved to be in general the best selection.

2.2.4 The β parameter

The β parameter intensifies the use of the heuristic information compared to the pheromone information accumulated so far. Obviously this parameter is completely dependent on the instance of the problem. In the literature there are not many studies that analyse its connection with the problem instance. Figure 2.4 shows an example of how this parameter can change from one instance to another. In Figure 2.4-left, the heuristic information alone is sufficient to find the best tour, so setting a big value of β that overwhelms the α parameter would be a good choice. This is an extreme case, since the nearest neighbour heuristic alone is sufficient to find the best solution. While in Figure 2.4-right, starting from the bottom cross point, the heuristic information allows us follow a wrong path

in finding the best solution. So, a smaller β is suggested. As mentioned, starting from the same node at every iteration is a common practice, as it is useful in practical implementations due to the resulting savings in computing time.

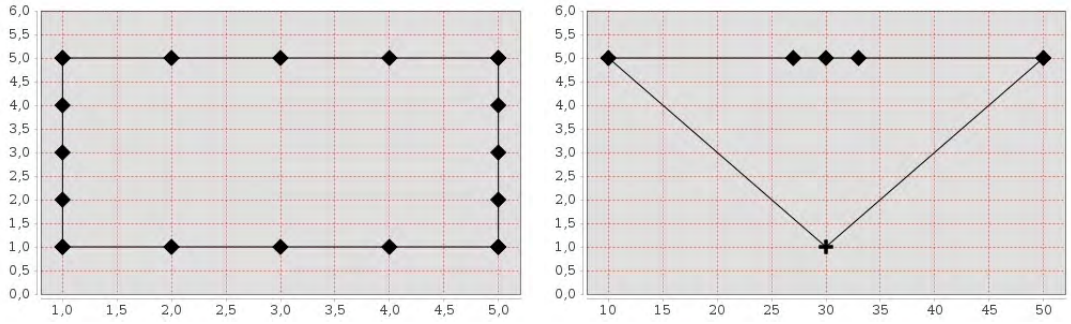

Figure 2.4: *Two example solutions with different best β values: in left graph, with $\beta = \infty$ we can easily reach the best solution for the TSP problem. Indeed the nearest neighbour heuristic is sufficient. In the right graph, starting from the bottom node, heuristic information makes finding the best solution harder.*

In [27] we can see some further details about the ratio β/α. This proportion determines the influence of the heuristic and pheromone information.

In the literature, $\beta = 2$ is commonly used as the default value.

2.2.5 The ρ parameter

The ρ parameter makes the pheromone evaporate as nature does with real pheromone. The higher ρ is, the higher the power of forgetting past knowledge. A ρ that is too high can lead to being trapped in a local optimum, while a ρ that is too small makes the algorithm take a long time before the accumulated knowledge becomes effective in the choice of arcs.

Obviously this parameter must be contained in the set $[0, 1]$. For a long runtime, a good choice seems to be 0.02 as described in [11], while other authors prefer 0.2 giving a shorter runtime.

In [27] a theoretical study of ρ is presented. Given the number of nodes of the problem instance, the authors relate ρ with the number of iterations. Since this study is strongly connected with the Max-Min AS variant, we propose its use in Section 2.4.

2.2.6 The $\Delta\tau_{i,j}$ function

The $\Delta\tau_{i,j}$ function defines the amount of pheromone to add at each iteration to the best solution found. There are no specially dedicated theoretical studies in the literature of this function, and it is commonly accepted to use the following function:

$$\Delta\tau_{i,j} := \begin{cases} Q \cdot l_{bs}^{-1} & if \quad (i,j) \in l_{bs} \\ 0 & otherwise \end{cases}$$

In the previous formula, $Q \in \mathbb{R}$ is normally set to 1 and l_{bs} is the length of the best solution found. The best solution found can be either the *iteration best* solution, l_{ib}, that is the best solution found at each iteration, or the *global best* solution, l_{gb}, that is the best solution found from the beginning of the algorithm. Both choices can be used, but the iteration best solution seems to give slightly better results compared with using the global best solution. This is due to the fact that using the global best solution makes the algorithm exploit the first found solutions, but at first reinforcing the exploration is preferred. It is also possible to mix the two strategies, for example allowing the algorithm to take the iteration best solution at the beginning and then to use the global best solution, or either to use the global best solution at some prefixed iterations (for example every $20 - 25$ iterations) and the iteration best solution otherwise.

2.2.7 On-line and Off-line adaptation of parameters

As seen before, successfully using of Ant System strongly depends on the parameters we described. Unfortunately, there is not a global best choice, as they are problem dependent. In the literature there are not many theoretical studies about the relationship between the problem instance and the parameters. As an example, in [12] the authors try to define two characteristics of the problems, depending on the distance between the nodes, and try to relate them with one or more parameters. Unfortunately they found no relation.

On the other hand, the literature contains plenty of experimental studies and algorithms that want to find the best combination of parameters for each problem instance. In [31] a meta-ACO algorithm is proposed, where the parameters themselves have associated a pheromone (as the arcs of the problem have), and each ant chooses the parameters at each iteration using these pheromone values. Other examples can be found in [17] where the authors use Particle Swarm Optimization to optimize the parameters, or in [37, 30] where genetic algorithms are used. These and other hybrid methods can be divided into two sets: first ones where each ant has its own parameters, while in the second set, the parameters are equal for all the ants in the colony. Note that in some methods, multiple instances of Ant Systems algorithms are running at the same time. In the literature these types of algorithms are called "on-line" adaptive algorithms, emphasizing the fact that the parameters can change during the running of the algorithm, contrasting to the "off-line" ones where the parameters are chosen before in a separate way.

A complete analysis of on-line versus off-line methods can be found in [28] and [29]. Furthermore, in [29] the authors propose an off-line method to set the parameters using the F-race package algorithm, described in [4, 5, 22].

All the experiments they did confirm that off-line methods perform better than on-line ones. Nevertheless, the use of adaptive methods seems

not to improve and even sometimes to worsen normal ACO algorithms. In both articles the authors emphasize that the use of these methods makes sense only if the following conditions arise:

- few parameters need to be adapted
- a long runtime
- make the starting configuration as near as possible to the final expected result, using heuristic knowledge to set initial parameters, rather than the values suggested in the literature
- use it only in the case of low performance of the standard algorithm
- use it only in the case of having a great heterogeneity in the set of instances to be tackled

2.2.8 Differences with real ants

Some differences between real and virtual ants are described in [11]. First of all, we must mention the obvious observation that real ants can not count on the heuristic function, hence $\beta = 0$.

Regarding the α parameter, different species of insects seem to have different values, most commonly $\alpha = 1$ or $\alpha = 2$. As noticed from studying Formula 2.1, these values can lead to some problems, depending on the initial condition. Indeed, real ants also experience problems, as shown with an experiment in [13] and mentioned in [11]. In this experiment, as we can see in Figure 2.5, we allow the ants to choose the only possible path to the food source. After 30 minutes, when we have the total convergence on this path and a good amount of pheromone deposited there, we add a shorter branch to the environment. As we can see in the right graph, in most cases ants continue following the long path. This is due not only to the α parameter but also to the ρ parameter that makes the pheromone

evaporate. In nature, the ρ value is not big enough to invert the situation in the time allowed by the experiment.

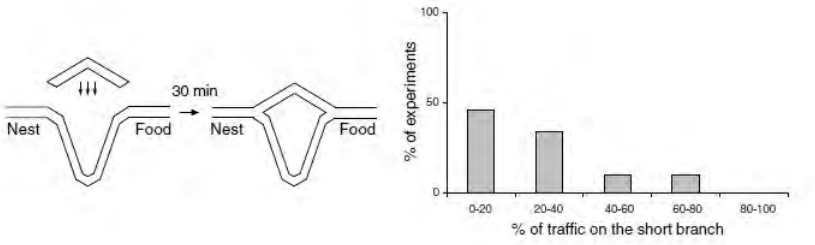

Figure 2.5: *Problem with real ants characterized by their value of α and the absence of heuristic information. Source: [11]*

Another important difference is the impossibility for real ants to know "how good" is the solution they have found. Virtual ants can compare among the solutions they find, and decide which is the best one, while real ants can not. This leads to two major consequences. The first one is that each real ant deposits the same amount of pheromone, while virtual ants deposit an amount proportional to the quality of the solution. The second consequence is that a real colony of ants uses a slightly different approach from the virtual one, that is, the method used by nature works based on the time the ants spent reaching the food source and coming back to the nest. In the same amount of time, the best path is covered more times, accumulating more pheromone. This approach has two limitations that virtual colonies do not have. Real colonies must be formed by more than one ant, and they need each ant to deposit the pheromone twice, on the way to the food source and on the way back. These limitations are not present in virtual colonies. Algorithms similar to the one used in nature have been implemented in the past, but they showed a lower performance with respect to current virtual colony versions.

2.3 Some useful practices

What we have seen till now, and what we will study in the following sections are basic ACO algorithms, that utilize just pheromone and a heuristic to find a solution. These are useful from a theoretical point of view, nevertheless some practices that seem to improve the results are nowadays part of ACO algorithms. We list here a few of them:

- alternate "iteration best" solution and "best so far" solution in the update function $\Delta\tau$; there are many ways to alternate them, one example is to use the best so far solution once every 20 or 25 iterations and iteration best solution otherwise
- reinitialize the algorithm when it seems to be stagnant and no new solutions are constructed; we reinitialize the pheromone matrix but maintain the global best solution tour in memory
- choose the next arc only from the best r arcs at each step; the algorithm can lose the possibility to reach the best solution, but the advantage in runtime makes this variant useful in real situations
- use a local search algorithm to improve the iteration best solution at every iteration; a list of some local search algorithms can be found in [11]; these algorithms look for a better solution near the iteration's solution; the utility of this method makes sense because usually good solutions are near the best one, as noted in [35] and described in more detail in [18], where we consider the distance between solutions as the number of arcs different in them

2.4 The Max-Min Ant System

The Max-Min Ant System is characterized by good experimental and also theoretical results that make it one of the most used ACO algorithms.

The main difference with respect to the pseudo-code in section 2.2 is the addition of a lower bound as well as an upper bound to the $\tau_{i,j}$ values, usually called τ_{min} and τ_{max}, from which the name of this variant arises. In particular we change the steps 18 and 19 of the pseudo-code to be the following:

18 $\forall i, j \ : \ \tau_{i,j} := max((1 - \rho)\tau_{i,j}, \tau_{min})$

19 $\forall i, j \ : \ \tau_{i,j} := min(\tau_{i,j} + \Delta\tau_{i,j}, \tau_{max})$

The purpose of these conditions is to overcome the usual problems of falling in local optimums, giving to each arc a minimum as a maximum probability of being chosen.

In [26] a critique of Max-Min AS is analysed, where the authors show that Max-Min AS spends more time updating the pheromone values than constructing solutions. This is due to the fact that the pheromone matrix becomes huge for big problem instances, and the time spent to limit the pheromone inside the range $[\tau_{min}, \tau_{max}]$ grows very fast. In their experiments the cost reaches more than 50% of time spent for little problems, reaching more than 88% for bigger ones.

Despite this critique, experiments confirm the usefulness of this variant in practice. Furthermore in articles such as [34, 14, 15] it is possible to find theoretical proofs of its convergence. In particular, in the literature three types of convergence are defined:

Definition 4. *Convergence in solution*

$$Pr(\omega(\tau(k)) \in \widetilde{\omega}) \to 1; \qquad k \to \infty$$

where k is the iteration number of the algorithm, $\tau(k)$ the pheromone matrix at iteration k, $\omega(\tau(k))$ a random path chosen using $\tau(k)$ and $\widetilde{\omega}$ the set of best paths.

Definition 5. *ϵ-convergence of a solution*
$\forall \epsilon \in]0,1[\ , \exists K(\epsilon) \in \mathbb{I}$ *such that*

$$Pr(\omega(\tau(k)) \in \widetilde{\omega}) \geq 1 - \epsilon; \qquad \forall k \geq K(\epsilon)$$

where k is the iteration number of the algorithm, $\tau(k)$ the pheromone matrix at iteration k, $\omega(\tau(k))$ a random path chosen using $\tau(k)$, $K(\epsilon)$ the minimum number of iterations from which the inequality holds, that depends on ϵ and $\widetilde{\omega}$ the set of best paths.

Definition 6. *Convergence in value*

$$Pr(f(\widetilde{\omega}(k)) = \widetilde{f}) \to 1; \qquad k \to \infty$$

where k is the iteration number of the algorithm, f the fitness function we want to maximize (or minimize), $\widetilde{\omega}(k)$ is the best solution found up until iteration k and $\widetilde{f}$ is the value of the fitness function calculated at the best solution of the problem.

In the mentioned articles, mathematical proofs are given that show that the Max-Min Ant System algorithm (from here on MMAS algorithm) is endowed with the ϵ-convergence of a solution, the convergence in value and that a variant of the MMAS algorithm, with dynamic ρ and τ_{min} also exhibits the convergence in solution. It is important to note that just a few of the ACO algorithms have been proven to have some of the convergence properties. This makes the MMAS algorithm interesting from a theoretical point of view.

As anticipated in section 2.2.5, an interesting study of the ρ parameter was given in [27]. We will repeat it here for convenience. As will be seen in detail in the next section, a common choice is $\tau_{max} = 1/\rho l_b$ where l_b is the length of the best solution. As we do not know this value, we will estimate it dynamically with l_{gb}, and at the start we use the solution of the nearest

neighbour heuristic, called l_{NN}. Also it is common to set $\tau_0 = 1/\rho l_{NN}$ and $\tau_{min} = [\tau_{max}(1-\sqrt[n]{0.05})]/[(n/2-1)\sqrt[n]{0.05}]$ where n is the number of nodes of the problem instance. From this data, we can say that an arc (i,j) never chosen after $\overline{k}$ iterations will have $\tau_{i,j} = (1-\rho)^{\overline{k}}\tau_0 = (1-\rho)^{\overline{k}}/\rho l_{NN}$. What the authors are interested in is calculating ρ such that after $\overline{k}$ iterations (with $\overline{k}$ a parameter the user can select) the probability to choose a never selected arc is minimum. This means that for iterations $k < \overline{k}$ we have $\forall i,j\ (i \neq j)\ :\ \tau_{i,j} > \tau_{min}$ and after $\overline{k}$ iteration we have $\tau_{i,j} = \tau_{min}$ for a generic (i,j) never selected. This leads to the following calculation:

$$\tau_{i,j}(\overline{k}) = \tau_{min};\ (1-\rho)^{\overline{k}}\frac{1}{\rho l_{NN}} = \frac{\frac{1}{\rho l_{NN}}(1-\sqrt[n]{0.05})}{(\frac{n}{2}-1)\sqrt[n]{0.05}};\ \rho = 1-\sqrt[\overline{k}]{\frac{(1-\sqrt[n]{0.05})}{(\frac{n}{2}-1)\sqrt[n]{0.05}}}$$

That last formula is used to demonstrate the following two propositions:

Proposition 1. *If, given n, $\overline{k}$ is such that $\rho \geq 1-\sqrt[\overline{k}]{\frac{(1-\sqrt[n]{0.05})}{(\frac{n}{2}-1)\sqrt[n]{0.05}}}$, then $\forall k > \overline{k}$ we have $\rho \geq 1-\sqrt[k]{\frac{(1-\sqrt[n]{0.05})}{(\frac{n}{2}-1)\sqrt[n]{0.05}}}$*

Proposition 2. *If, given $\overline{k}$, n is such that $\rho \geq 1-\sqrt[\overline{k}]{\frac{(1-\sqrt[n]{0.05})}{(\frac{n}{2}-1)\sqrt[n]{0.05}}}$, then $\forall m\ :\ 3 \leq m < n$ we have $\rho \geq 1-\sqrt[k]{\frac{(1-\sqrt[m]{0.05})}{(\frac{m}{2}-1)\sqrt[m]{0.05}}}$*

The relationship between the three parameters allows us to calculate a ρ, given the size of the problem instance n, such that after $\overline{k}$ iterations the "bad" arcs have the minimum probability to be chosen. Estimating the iterations, given the runtime, we are able to calculate the best ρ for our MMAS algorithm. As an example, for a problem of size 1000, if we set $\overline{k} = 100$, we would choose $\rho \sim 0.1$. Moreover the two propositions let us calculate a ρ value for a set of different instances, considering just the one with the biggest size.

2.4.1 τ_{max}, τ_{min} and τ_0 values

The τ_{max} and τ_{min} parameters are the innovative idea that MMAS implements. So, a detailed study of their values is required. The main result regarding these parameters can be found in [35]. Here we can find the motivation of the values mentioned in the previous section. Intuitively, the importance of these parameters is to solve the problem of being trapped in local optimum, giving a minimum probability of being chosen to each arc. This improves the exploration.

Starting from τ_{max}, we want to be able to deposit all the possible pheromone on each arc, supposing the arc to be chosen at each iteration. So, remembering that $0 < \rho < 1$ we will have:

$$max(\tau_{i,j}) = \sum_{k=1}^{k_{max}} (1-\rho)^{k_{max}-k} \frac{1}{f(l_b)} + (1-\rho)^{k_{max}} \tau_0 \rightarrow \frac{1}{\rho f(l_b)} \qquad k_{max} \rightarrow \infty \tag{2.2}$$

As already mentioned, we will estimate dynamically l_b with l_{gb} (or l_{NN} at start). As can be seen, setting this τ_{max} value is identical to not setting it at all. But what it is useful for, is to have a value from which a lower bound for τ_{min} can be calculated. Finally, the important value is the proportion τ_{max}/τ_{min}.

Continuing the study of τ_{min}, let us call *MMAS-convergence* the following situation: the best solution found at the moment has all arcs with τ_{max} pheromone while others arcs have τ_{min} (that we have to define now). Let us call p_{best} the probability of choosing the best solution. p_{best} means choosing at each step the (only) arc with the maximum pheromone, and we call this probability p_{dec}. We can simplify the problem assuming that p_{dec} is constant at each step, so that $p_{best} = p_{dec}^n$, or equivalently $p_{dec} = \sqrt[n]{p_{best}}$. Notice that on average, we must choose between $n/2$ arcs. This leads to:

$$p_{dec} = \frac{\tau_{max}}{\tau_{max} + (\frac{n}{2} - 1)\tau_{min}}$$

Solving for τ_{min}:

$$\tau_{min} = \frac{\tau_{max}(1 - p_{dec})}{(\frac{n}{2} - 1)p_{dec}} = \frac{\tau_{max}(1 - \sqrt[n]{p_{best}})}{(\frac{n}{2} - 1)\sqrt[n]{p_{best}}} \tag{2.3}$$

In this equation we can observe that if $p_{best} = 1$ then $\tau_{min} = 0$. On the other hand, if p_{best} is too small, $\tau_{min} > \tau_{max}$, so in this case we will set $\tau_{min} = \tau_{max}$, however this implies not using the pheromone information at all.

Thus, in the above discussion we fixed a τ_{max} value, and set a relationship between p_{best} and τ_{min}, hence also between p_{best} and τ_{max}/τ_{min}. Note that in this theoretical consideration the heuristic information is not used at all, even if it is commonly used in the literature as well as in the experiments done in the article itself.

The above discussion lets the user choose τ_{min} based on the p_{best} wanted. After some few experiments, the authors propose to set $p_{best} = 0.05$, that seems to be the average of the best choices for the problem instances they tried.

Regarding the τ_0 value, we can find some considerations in the same article. Actually, we can set τ_0 to any value since it will situate in the range $[\tau_{min}, \tau_{max}]$ in the second iteration. So, it is more interesting to study the effect of this two extremes. As seen before, if we set $\tau_0 = \tau_{max}$ (at the beginning, we can achieve this result by using a value big enough, or $1/\rho f(l_{NN})$, since we do not know τ_{max}), then at each iteration the pheromone on the arcs that have not been chosen will decrease by ρ. On the other hand, if we set $\tau_0 = \tau_{min}$ then the difference at first iteration will be $\frac{(1-\sqrt[n]{p_{best}})}{(\frac{n}{2}-1)\sqrt[n]{p_{best}}}$, which is much bigger than ρ. In conclusion, to reinforce exploration at the beginning of the algorithm, empirical experiments

confirmed $\tau_0 = \tau_{max}$ as the best choice.

Chapter 3

The Problem of ACO Parameters

As we have seen in Sections 2.2 and 2.4, ACO algorithms are endowed with parameters that can strongly change the effectiveness of the algorithm to solve problems, for instance the TSP. The majority of the studies of these parameters seem to be experimental, using some set of instances of TSP. Furthermore, meta-algorithms which try to select a good combination of them, are not so effective once they are taken out from the environment where they have been constructed.

The theoretical studies of ACO algorithms, usually do not relate parameters. What we propose is to continue the study of [24] analyzing the effect of other parameters, and give a relationship between them using a reasoning similar to [35]. Then in the next chapter we will show some experiments. We are convinced that a further understanding of parameters can help ACO users set efficient parameter combinations at the beginning of the ACO algorithm using heuristic knowledge of the TSP instance they want to solve. As far as we can see in the literature, nowadays the human understanding of the problem is still essential.

3.1 Stability analysis

We will continue the study described in Section 2.2.3 with the graph in Figure 2.2, but considering a more general case and analyse again the stability. From the probability function $\forall i = 1, 2 \ : \ p_i = \tau_i^\alpha \eta_i^\beta / (\tau_1^\alpha \eta_1^\beta + \tau_2^\alpha \eta_2^\beta)$ and the pheromone update $\forall i = 1, 2 \ : \ \tau_i := (1 - \rho)\tau_i + Q \cdot q(l_i)$ if l_i is chosen, where $q(i_i)$ is a quality function $q \ : \ \mathbb{R} \to \mathbb{R}$, we have the condition:

$$\forall i = 1, 2 \ : \ \rho \cdot \tau_i = Q \cdot q(l_i) \cdot m \cdot p_i^* \tag{3.1}$$

Following the calculation we will have:

$$\frac{p_1^*}{p_2^*} = \frac{\tau_1 \cdot q(l_2)}{\tau_2 \cdot q(l_1)}; \quad \frac{\tau_1^\alpha \cdot \eta_1^\beta}{\tau_2^\alpha \cdot \eta_2^\beta} = \frac{\tau_1 \cdot q(l_2)}{\tau_2 \cdot q(l_1)}; \quad \frac{\tau_1^{\alpha-1} \cdot \eta_1^\beta}{\tau_2^{\alpha-1} \cdot \eta_2^\beta} = \frac{q(l_2)}{q(l_1)};$$

$$\frac{\tau_1^{1-\alpha} \cdot \eta_1^{-\beta}}{\tau_2^{1-\alpha} \cdot \eta_2^{-\beta}} = \frac{q(l_1)}{q(l_2)}; \quad \frac{\tau_2}{\tau_1} = \left(\frac{q(l_2) \cdot \eta_2^\beta}{q(l_1) \cdot \eta_1^\beta} \right)^{\frac{1}{1-\alpha}}$$

Substituting this proportion in the probability function as before, we reach the following formula:

$$p_1^* = \frac{\tau_1^\alpha \cdot \eta_1^\beta}{\tau_1^\alpha \cdot \eta_1^\beta + \tau_2^\alpha \cdot \eta_2^\beta} = \left(\frac{\tau_1^\alpha \cdot \eta_1^\beta + \tau_2^\alpha \cdot \eta_2^\beta}{\tau_1^\alpha \cdot \eta_1^\beta} \right)^{-1} = \left(1 + \frac{\tau_2^\alpha \cdot \eta_2^\beta}{\tau_1^\alpha \cdot \eta_1^\beta} \right)^{-1} =$$

$$= \left(1 + \left(\frac{q(l_2) \cdot \eta_2^\beta}{q(l_1) \cdot \eta_1^\beta} \right)^{\frac{\alpha}{1-\alpha}} \frac{\eta_2^\beta}{\eta_1^\beta} \right)^{-1} = \left(1 + \left(\frac{q(l_2)}{q(l_1)} \right)^{\frac{\alpha}{1-\alpha}} \cdot \left(\frac{\eta_2}{\eta_1} \right)^{\beta\left(\frac{\alpha}{1-\alpha}+1\right)} \right)^{-1} =$$

$$= \left(1 + \left(\frac{q(l_2)}{q(l_1)} \right)^{\frac{\alpha}{1-\alpha}} \cdot \left(\frac{\eta_2}{\eta_1} \right)^{\left(\frac{\beta\alpha}{1-\alpha}+\frac{\beta(1-\alpha)}{1-\alpha}\right)} \right)^{-1} =$$

$$= \left(1 + \left(\frac{q(l_2)}{q(l_1)}\right)^{\frac{\alpha}{1-\alpha}} \cdot \left(\frac{\eta_2}{\eta_1}\right)^{\frac{\beta}{1-\alpha}}\right)^{-1} = \left(1 + \left(\left(\frac{q(l_2)}{q(l_1)}\right)^{\alpha} \cdot \left(\frac{\eta_2}{\eta_1}\right)^{\beta}\right)^{\frac{1}{1-\alpha}}\right)^{-1} \tag{3.2}$$

From equation 3.2 we can make some conclusions, keeping in mind that $l_1 < l_2$.

3.1.1 Variation of the parameter ρ

First of all notice that the ρ parameter does not take any role in equation 3.2, since it was replaced at the beginning. Indeed it is applied to all the arcs, regardless of whether the arc is chosen or not. So it does not change the probability function, since it depends on the ratio of τ_1 and τ_2, beyond other elements. In the mathematical formula, considering the probability function at iteration $k+1$, before applying the $\Delta\tau$ function we have:

$$p_1^{(k+1)^-} = \frac{((1-\rho)\tau_1)^\alpha \cdot \eta_1^\beta}{((1-\rho)\tau_1)^\alpha \cdot \eta_1^\beta + ((1-\rho)\tau_2)^\alpha \cdot \eta_2^\beta} =$$

$$= \frac{(1-\rho)^\alpha}{(1-\rho)^\alpha} \cdot \frac{\tau_1^\alpha \cdot \eta_1^\beta}{\tau_1^\alpha \cdot \eta_1^\beta + \tau_2^\alpha \cdot \eta_2^\beta} = \frac{\tau_1^\alpha \cdot \eta_1^\beta}{\tau_1^\alpha \cdot \eta_1^\beta + \tau_2^\alpha \cdot \eta_2^\beta} = p_1^k$$

What that previous equation means is that we actually could take away ρ from the list of parameters, obviously modifying a bit the algorithm. For example we could leave τ_{max} increase indefinitely and use a dynamic τ_{min} value that increases in relation to τ_{max}. Nevertheless this causes other practical problems, as for example the problem of the increased size of τ_{max}, the implementation of the process that modifies τ_{min} and the loss of the simplicity of the method.

In Figure 3.1 we can see a numerical experiment that shows what we have just demonstrated. Figure 3.1-left with $\rho = 0.3$ is identical to the

one on the right with $\rho = 0.7$.

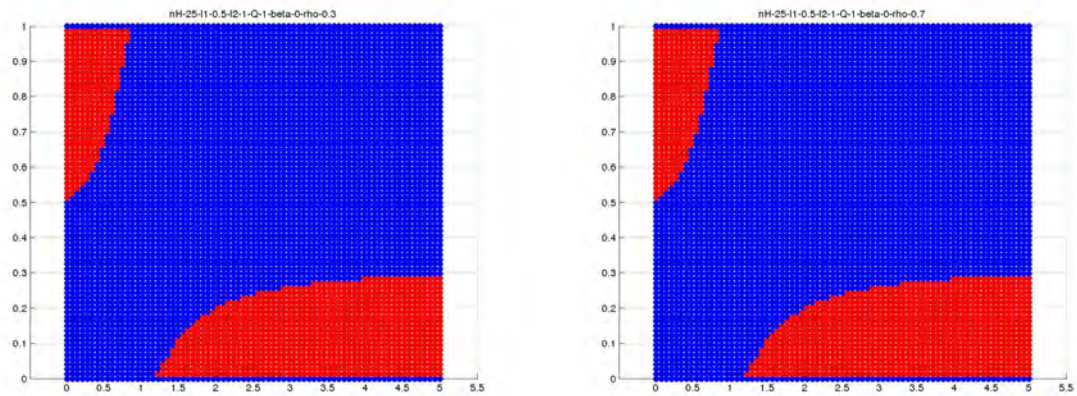

Figure 3.1: *A numerical study of the stability of the probability function, using* 25 *ants,* $Q = 1$, $l_1 = 0.5$, $q(l_1) = \eta_1 = 2$, $l_2 = 1$, $q(l_2) = \eta_2 = 1$, $\beta = 0$, $\rho = 0.3$ *on the left and* $\rho = 0.7$ *on the right,* α *on the x-axis. Red points represent the probability decrease and blue points the probability increase with iterations. Both graphs are identical, showing the null influence of* ρ *in this case.*

3.1.2 Variation of the parameter Q

Continuing with the Q parameter, we mentioned that we would experiment with different values. In Figure 3.2 we show the same study with $Q = 1$ on the left and $Q = 3$ on the right. As we can see there is no difference between the two pictures. This means that Q is also a parameter that does not influence the stability analysis we are discussing. This is due to the fact that Q increases the difference $p_1^{k+1} - p_1^k$ but it does not influence its sign. It leads to a more rapid convergence intensifying the pheromone update on the chosen arcs at each iteration, but being a constant it does

not depend on the quality of the solution. As we already said, in literature using $Q = 1$ is common and no further studies about it are exhibited.

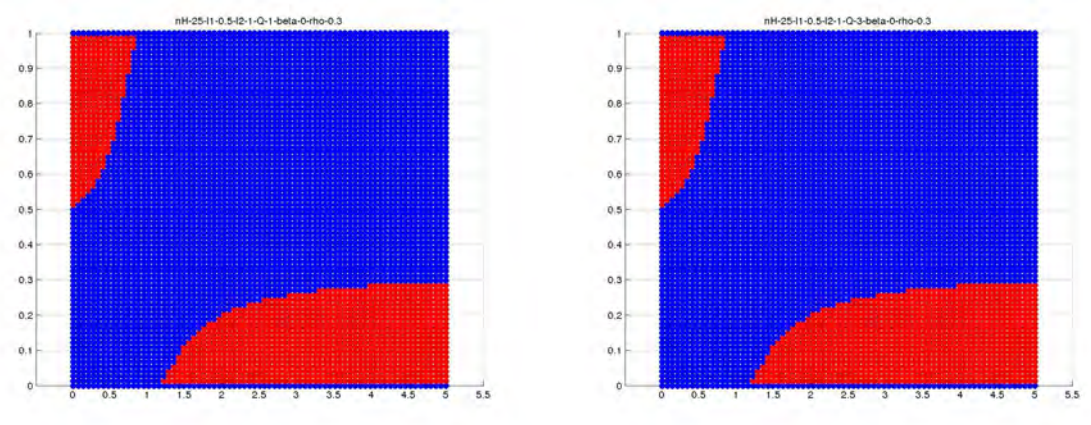

Figure 3.2: *A numerical study of the stability of the probability function, using* 25 *ants,* $l_1 = 0.5$, $q(l_1) = \eta_1 = 2$, $l_2 = 1$, $q(l_2) = \eta_2 = 1$, $\beta = 0$, $\rho = 0.3$, $Q = 1$ *on the left and* $Q = 3$ *on the right,* α *on the x-axis. Red points represent the probability decrease and blue points the probability increase with iterations. Both graphs are identical, showing the null influence of* Q *in this case.*

3.1.3 Variation of the parameter α

The next parameter we discuss is α. As we saw in Section 2.2.3, this parameter has a strange effect, due to the singularity at $\alpha = 1$. In Figure 3.3, where two numerical experiments of the theoretical Figure 2.2 are reproduced, we see that also in these cases, with $\alpha = 1$ we have a singularity with a single fix-point at $p_1 = 1$.

When we have $\alpha < 1$, the bigger the value of α, the bigger the influence of the quality function due to the $\alpha/(1-\alpha)$ exponent in the term $q(l_2)/q(l_1)$

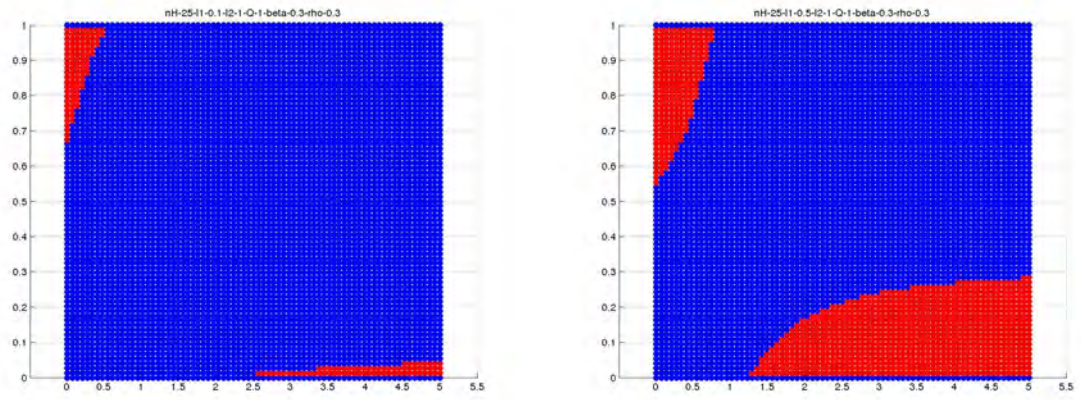

Figure 3.3: *A numerical study of the stability of the probability function, using* 25 *ants,* $\rho = 0.3$, $\beta = 0.3$, $l_1 = 0.1$, $q(l_1) = \eta_1 = 10$, $l_2 = 1$, $q(l_2) = \eta_2 = 1$ *on the left picture and* $l_1 = 0.5$, $q(l_1) = \eta_1 = 2$, $l_2 = 1$, $q(l_2) = \eta_2 = 1$ *on the right one,* α *on the x-axis. Red points represent the probability decrease and blue points probability increase with iterations.*

and the bigger the influence of the heuristic element because of the same reason. As already said in section 2.2.3, the choice of $\alpha < 1$ ensures the convergence to the best solution as well as a certain degree of diversity in the solution. In theoretical studies, it is important to have a balanced set of solutions where we can choose the best one, since it increases the exploration and we need to find the best solution just once. Unfortunately this choice also increases the time that the algorithm needs to find it, thus it could be a good option only in the case that the computational time is enough in respect to the given size of the problem instance. As mentioned in Section 2.3 with the use of local search algorithms, the configuration of local optimum is not by chance, as shown in [35] or in [18] with further details explanation, but they concentrate in a small region around the global best solution. Moreover, the closer they are with respect to the global best, the better the solution reached. In Figure 3.4 we can see the correlation

we are talking about. These observations demonstrate that exploration and exploitation of the solutions do not have the same importance, and improving exploitation gives in general more promising results.

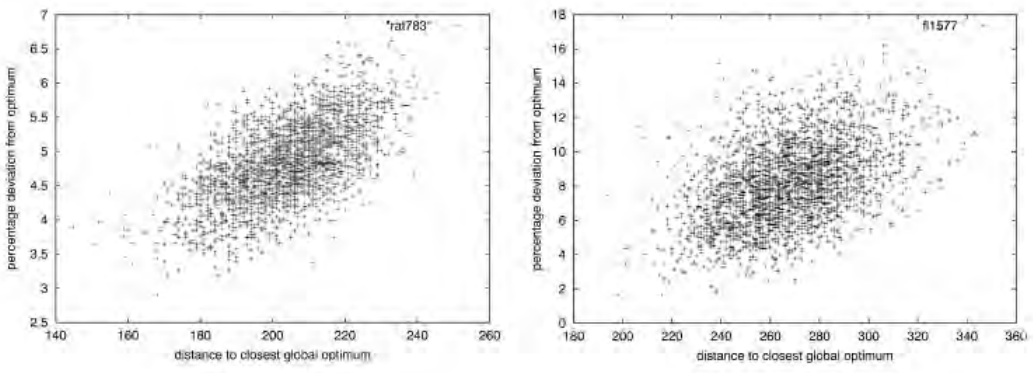

Figure 3.4: *Locations of solutions when modifying the best one. The distance between two solutions is the number of arcs on which they differ. The two pictures, for the problems* "rat783" *on the left and* "fl1577" *on the right, show a correlation between the distance from the best solution and the quality of the solution itself. Source: [35]*

On the other side of the singularity, where $\alpha > 1$, the exponential term $1/(1-\alpha)$ becomes negative and the behaviour changes radically. As we already said, the middle fixed point becomes unstable. Letting the α value grow, we can also notice that the heuristic element loses importance, eventually reaching 1 since $\beta/(1-\alpha) \to 0 \ : \ \alpha \to \infty$. Moreover as we can see from Figure 3.3, there is an asymptote $\alpha/(1-\alpha) \to -1 \ : \ \alpha \to \infty$ where the importance of the quality element is flattened to $q(l_1)/q(l_2)$. The asymptotic behaviour is:

$$p_1^{*,\alpha\to\infty} = \frac{1}{\left(1 + \left(\frac{q(l_1)}{q(l_2)}\right)\right)} \qquad : \qquad \alpha \to \infty$$

In the case that we are using $\alpha > 1$, in situations where we are more interested in reducing the computational time rather than finding the best solution, $p_1^{*,\alpha\to\infty}$ is the upper bound of the risk of finding a bad solution. Since it depends on the ratio $q(l_1)/q(l_2)$, it could be possible to calculate this risk. For example:

$$\left.\begin{array}{rcl} q(l_1) & = & 2 \\ q(l_2) & = & 1 \end{array}\right\} \Rightarrow p_1^{*,\alpha\to\infty} = 0.33 \quad ; \quad \left.\begin{array}{rcl} q(l_1) & = & 10 \\ q(l_2) & = & 1 \end{array}\right\} \Rightarrow p_1^{*,\alpha\to\infty} = 0.09$$

Thus, if we have $p_1 > p_1^{*,\alpha\to\infty}$ after first iterations of the algorithm (when the selection of the arcs depends mainly on the heuristic function), then the algorithm will converge to $p_1 = 1$ (recall that in this study we are considering $l_1 < l_2$). So we can say that, qualitatively speaking, there is a bigger probability of obtaining smaller errors, and a lower probability of obtaining bigger ones. This reasoning could provide a base theory for real situations where α values bigger than 1 could reach statistically good results, given a very short computational time in respect to the size of the problem instance. Nevertheless a deeper study of this behaviour mixed with other practical tools, as the ones seen in Section 2.3, is required.

3.1.4 Variation of the parameter β

Continuing the analysis of the β parameter, we see that it increases the heuristic element in the formula 3.2. We can observe this behaviour also in Figure 3.5. Its influence is emphasized on the left side of the α singularity while on the right side, confirming the result reached in the previous study, the heuristic function maintains its importance just in the case $\alpha \approx \beta$. As far as we understood from the Equation 3.2, we are not completely free to choose the α value since it can drastically change the behaviour of the algorithm, due to the singularity inside the term $1/(1-\alpha)$. Instead we

can use the β value to handle the steady state of the probability p_1^* without worrying about changing this behaviour. Obviously it depends on the quality function and the heuristic function, but as we will see now, it is a common choice to set $q(l_i) = \eta_i$.

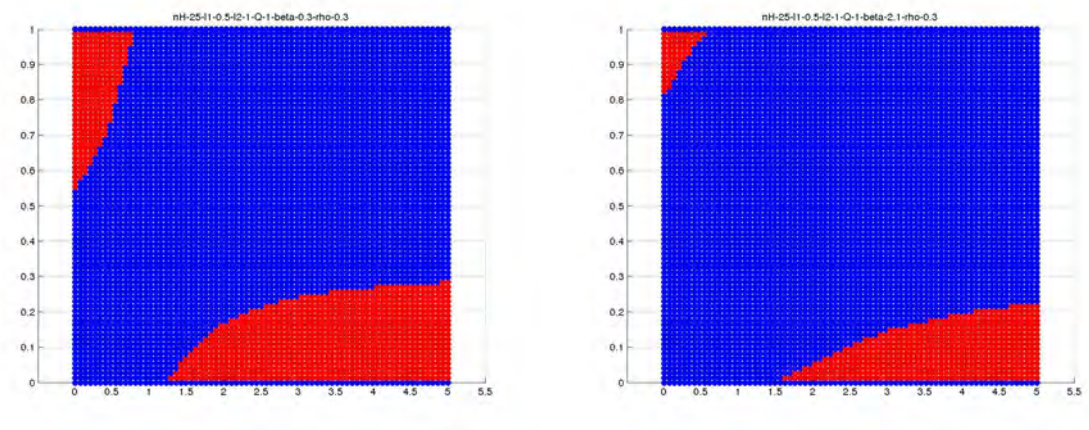

Figure 3.5: *A numerical study of the stability of the probability function, using* 25 *ants,* $l_1 = 0.5$, $q(l_1) = \eta_1 = 2$, $l_2 = 1$, $q(l_2) = \eta_2 = 1$, $Q = 1$, $\rho = 0.3$, $\beta = 0.3$ *on the left and* $\beta = 2.1$ *on the right,* α *on the x-axis. Red points represent the probability decrease and blue points the probability increase with iterations. The two graphs show the influence of* β *in this case.*

3.1.5 Quality and heuristic functions $q(l_i)$ and η_i

With respect to the quality and the heuristic functions, in the literature there are not many studies about them and a common value is used, giving more emphasis to the study of the parameters α and β. The common choice appears to be reasonable and functional for the problem in question, the TSP, and we agree with it:

$$\eta_i := \frac{1}{l_i} \quad ; \quad q(l_i) := \frac{1}{l_i}$$

With this choice, it is possible to simplify the formula 3.2 as follows:

$$p_1^* = \left(1 + \left(\frac{q(l_2)}{q(l_1)}\right)^{\frac{\alpha}{1-\alpha}} \cdot \left(\frac{\eta_2}{\eta_1}\right)^{\frac{\beta}{1-\alpha}}\right)^{-1} =$$

$$= \left(1 + \left(\frac{l_1}{l_2}\right)^{\frac{\alpha}{1-\alpha}} \cdot \left(\frac{l_1}{l_2}\right)^{\frac{\beta}{1-\alpha}}\right)^{-1} = \left(1 + \left(\frac{l_1}{l_2}\right)^{\frac{\alpha+\beta}{1-\alpha}}\right)^{-1}$$

Here it is easy to note the effect of the α and β parameters. Only α appears in the denominator of the exponent, that is why it leads to a more interesting study. With the last formula in mind, given $q(l_i) = \eta_i$, it is easy to manage the parameter α and β jointly: given a fixed α, in particular deciding whether $\alpha < 1$ or $\alpha > 1$, we can regulate the probability p_1^* using the heuristic function.

3.1.6 Number of ants, τ_{max}, τ_{min} and differences between theoretical and real situations

In the theoretical study we did above, the number of ants in Formula 3.1 is replaced and it does not play any role. This is also the case with Q and ρ, as can be seen in Figure 3.6. This effect is due to some implicit causes in the theoretical study that are not true when we implement the real algorithm.

First of all, to match real implementations, we should consider just the one-ant situation with the iteration best updating method. In Formula 3.1 we are using the fact that each ant leaves pheromone on the arcs at each iteration. As we already said, this leads to very poor performance

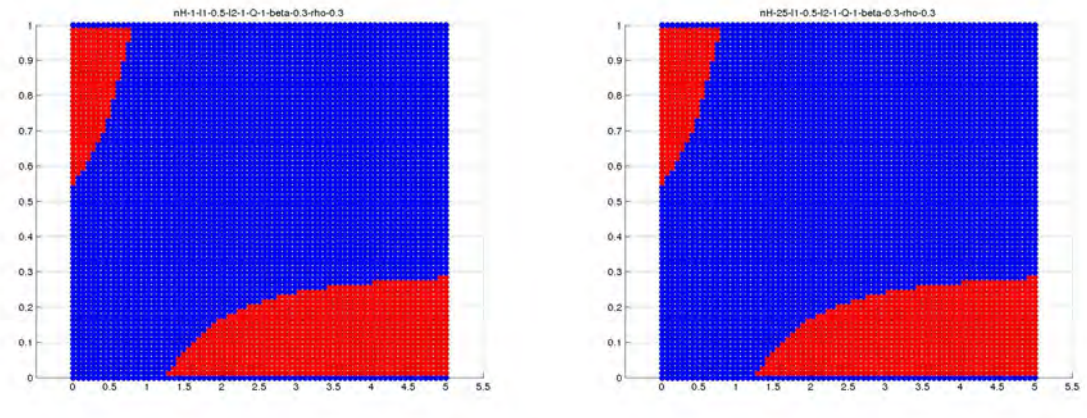

Figure 3.6: *A numerical study of the stability of the probability function, using* $l_1 = 0.5$, $q(l_1) = \eta_1 = 2$, $l_2 = 1$, $q(l_2) = \eta_2 = 1$, $\beta = 0.3$, $\rho = 0.3$, $Q = 1$, 1 *ant on the left and* 25 *on the right,* α *on the x-axis. Red points represent the probability decrease and blue points the probability increase with iterations. Both graphs are identical, showing the null influence of the number of ants in this case.*

and currently all algorithms let just one ant update pheromone. Hence the theoretical study makes sense just when only using one ant.

Leaving aside this consideration, there is another difference we have observed. In the theoretical formula the number of ants is a real variable while in the real implementation it is not. This means that we have considered the term $m \cdot p_i$ to be in the real numbers set $\mathbb{R}$, while in practice it must lie in the set of integer numbers $\mathbb{I}$. Considering for example the situation where $m = 1$, $p_1 = 0.1$, $\beta = 0$, $\alpha = 0.5$, as we see in Figure 3.7-left, the probability should go up until reaching the stable fixed-point p_1^* as we saw in the previous section. Nevertheless, there is a probability of 90% that the ant will go down, following the l_2 path. In this case, having just one ant, the pheromone on l_2 will be updated, while the pheromone

on l_1 evaporates. Depending on the quality function, it is possible that the unstable fixed-point $p_1 = 0$ is reached, causing the ant to always follow the bad arc in the future. So it is indeed possible to reach unstable fixed-points. The fewer the number of ants and the lower the p_1 value, the bigger the probability of reaching a bad case in which only the bad solution is considered.

To show the intuition of this reasoning, we repeated the numerical experiment in Figure 3.7-right where we used 10 ants and $\beta = 0$ and we added a "round" function on the term $m \cdot p_i$ to simulate the implementation. Here we can see that, even if the general trend is quite similar, there are differences, the most important ones in the top and bottom lines where the evolution of p_i has changed. In the bottom we can see how it is possible to reach the $p_1 = 0$ equilibrium and in top the $p_1 = 1$ one. In next section we will analyse in more detail this behaviour.

We finish this study analysing the effect of the τ_{max} and τ_{min} parameters. As shown in Figure 3.8, the proportion τ_{max}/τ_{min} and the conditions described at the beginning of Section 2.4, create unattainable regions (the black regions in the figure) that improve the exploration of the algorithm and partially eliminate the problem we have seen in Figure 3.7. The areas of these regions depends on the two parameters that we will study carefully in next section.

As we discussed, α and β have a different role from τ_{max} and τ_{min} values. While the first two parameters move the p_1^* probability, the last two give upper and lower bounds to it. Keeping in mind this relation is useful in finding a good choice for each parameter in a joint manner when we are setting the parameters values at the beginning of the algorithm. In the case of setting $\alpha > 1$, the use of τ_{max} and τ_{min} allows the management of the risk of choosing bad solutions. A detailed study of these situations in the future would be very interesting.

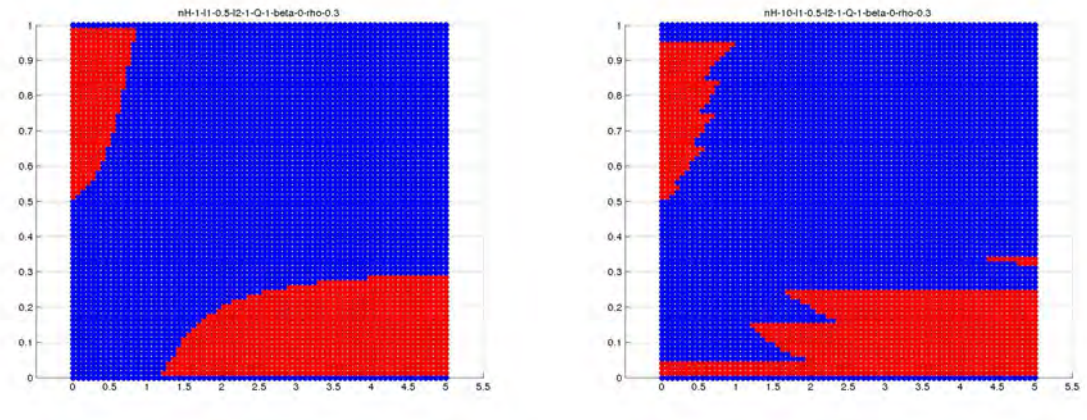

Figure 3.7: *On the left a numerical study of the stability of the probability function, using* 1 *ant,* $l_1 = 0.5$, $q(l_1) = \eta_1 = 2$, $l_2 = 1$, $q(l_2) = \eta_2 = 1$, $\beta = 0$, $\rho = 0.3$, $Q = 1$, α *on the x-axis. On the right the same study using* 10 *ants,* $l_1 = 0.5$, $q(l_1) = \eta_1 = 2$, $l_2 = 1$, $q(l_2) = \eta_2 = 1$, $\beta = 0$, $\rho = 0.3$, $Q = 1$, α *on the x-axis, adding a "round" function to simulate the case of no-divisible ants. Red points represent the probability decrease and blue points the probability increase with iterations.*

3.2 A relation among the number of ants, τ_{max}, τ_{min}, α and β parameters

In Section 2.4.1, we mentioned some reasoning about the τ_{max} and τ_{min} parameters taken from [35]. The result shows the suggested values of them described in Equations 2.2 and 2.3 that we will repeat here for convenience:

$$\tau_{max} = \frac{1}{\rho f(l_b)} \; ; \; \tau_{min} = \frac{\tau_{max}(1 - \sqrt[n]{0.05})}{(\frac{n}{2} - 1)\sqrt[n]{0.05}} \; ; \; \frac{\tau_{max}}{\tau_{min}} = \frac{(\frac{n}{2} - 1)\sqrt[n]{0.05}}{(1 - \sqrt[n]{0.05})} \quad (3.3)$$

Where n is the number of nodes in the problem and $0.05 = p_{best}$ is the probability to choose the best solution once the MMAS algorithm has

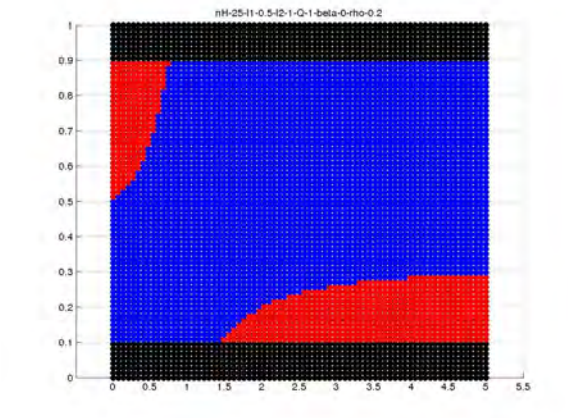

Figure 3.8: *A numerical study of the stability of the probability function, using* 25 *ant,* $l_1 = 0.5$, $q(l_1) = \eta_1 = 2$, $l_2 = 1$, $q(l_2) = \eta_2 = 1$, $\beta = 0$, $\rho = 0.2$, $Q = 1$, α *on the x-axis. Red points represent the probability decrease and blue points the probability increase with iterations. The black regions are out of reach do to the effect of the* τ_{max}/τ_{min} *proportion.*

MMAS-converged. In this context the *MMAS-convergence* property means that all the arcs have τ_{min} pheromone except the arcs in the best solution which have τ_{max}.

Without disagreeing with the authors, we noticed that the suggested parameter $p_{best} = 0.05$ is the average of the best values they found in their tests, but it is not the best value in general. This means that a greater understanding of the proportion τ_{max}/τ_{min} could let users understand better the behaviour of the MMAS algorithm and customizing these parameters if it is needed. Moreover, Formula 3.3 does not take in consideration the number of ants.

Reflecting on the problems of the number of ants described in Section 3.1.6, we have already said that, when we are using a "few" ants, then there are regions where the unstable fixed-points become, in some sense,

stable, as shown in Figure 3.7. The term "few" depends on the proportion τ_{max}/τ_{min}, which in turn depends on the number of nodes n of the problem. In Figure 3.8 we saw that τ_{max} and τ_{min} create unattainable regions near the unstable fixed-points. What we want to analyse is *how much* the two types of regions are above each other. Then we approached the problem of τ_{max} and τ_{min} values from a different point of view than the one the authors considered in [35].

First, we calculated the probability of an arc not being chosen by any of the m ants. Here we show the calculation:

$$p_i^{not} = (1 - p_i)^m = \left(1 - \frac{\tau_i^\alpha \cdot \eta_i^\beta}{\sum_{j=1}^{n}(\tau_j^\alpha \cdot \eta_j^\beta)}\right)^m$$

$$p_{bad}^{not} = (1 - p_{bad})^m = \left(1 - \frac{\tau_{min}^\alpha \cdot \eta_{min}^\beta}{\tau_{min}^\alpha \cdot \eta_{min}^\beta + \sum_{j=1, j\neq min}^{n}(\tau_j^\alpha \cdot \eta_j^\beta)}\right)^m \tag{3.4}$$

In real situations the probability p_{bad}^{not} can be really high, for instance it can be so high that it can be considered 1 in a double precision machine. It is important to notice that in this situation the corresponding arc will not be considered at all as a concrete possibility and will never be chosen. Hence the τ_{max} and τ_{min} parameters lose their usefulness. Formula 3.4 depends on α, β, n, m, τ_{max} and τ_{min}, and we want to see how to manage them to improve p_{bad}^{not}. In this context, the arcs called *bad arcs*, could be just arcs that we never considered until now, and are not necessarily "bad". It could be the case, as we said several times, that in initial iterations the arcs are chosen principally using the heuristic function rather than the pheromone value.

Notice that the number of nodes n is fixed from the problem instance and we can not change it. When the algorithm starts, we must choose

between a great number of arcs, that in the problems we will consider it will vary from 100 to 5934. Since we start always from the same node, as is common practice, there will be some arcs starting from this node that will never be chosen. We assume this is an internal problem of the implementations, and we do not try to solve it. At each step, the number of available arcs decreases, as we consider just the nodes that are not chosen yet. It makes sense to repeat the following reasoning using $n/2$ which is the average of the number of arcs we must choose from at each step. Anyway, as n is an instance depending value, we center our attention on the graph in Figure 2.2 to simplify the reasoning. Hence we have in the following $n = 2$.

We already discussed in Section 3.1 the α and β parameters and the consequences of their variation. Here we will set $\alpha = 1$ and $\beta = 0$ and we will not change them, although it is possible to repeat the following reasoning with the values the user wants to set in his implementation.

Using the above conditions, we reach a simpler equation that depends only on m, τ_{max} and τ_{min}. Considering the probability of not choosing the bad arc, once the MMAS algorithm has converged, we have the following:

$$p_{bad}^{not} = (1 - p_{bad})^m = \left(1 - \frac{\tau_{min}}{\tau_{min} + \tau_{max}}\right)^m \tag{3.5}$$

Since it is possible that not every arc with τ_{min} pheromone has actually been chosen in past iterations, it is reasonable to give them a minimum probability to be chosen in future. Finally, that is indeed the idea of the MMAS algorithm. In the following we respect the value of τ_{max} as in Formula 3.3, and we will study only the value of τ_{min}.

In Table 3.1 we can see how difficult is that at least one ant chooses an arc with τ_{min} pheromone in one iteration. Considering the number of iterations that is possible to do in half an hour, summarized in Table 3.2, let us say that in big problems these arcs are probably never chosen at

Table 3.1: *Some values of p_{bad}^{not} for some graphs with different numbers of arcs, with different numbers of ants (m) and considering just the selection between two arcs as in formula 3.5. We used $\alpha = 1$ and $\beta = 0$.*

Graph	τ_{max}/τ_{min}	$m = 1$	$m = 25$
kroA100	≈ 1660	0.99939	0.98506
rat783	≈ 102261	0.99999	0.99975
pr2392	≈ 954770	0.99999	0.99997
rl5934	≈ 5876593	0.99999	0.99999

Graph	$m = 250$	$m = 1000$	$m = 10000$
kroA100	0.86027	0.54771	0.00242
rat783	0.99755	0.99026	0.90684
pr2392	0.99973	0.99895	0.98958
rl5934	0.99995	0.99982	0.99829

all. This is a really optimistic calculation without considering the heuristic function and comparing just two arcs, while in real situations, with a double precision machine, we reach $p_{bad}^{not} = 1$, hence the arc is completely excluded from the possibilities.

Finally, a carefully study of the proportion τ_{max}/τ_{min} could improve the performance of the algorithm. Given a fixed computational time, the variables we can manage are τ_{max}/τ_{min} and the number of ants m. A trade-off between them is required, since, as shown in Table 3.2 and in Table 3.3, a greater number of ants drastically reduces the number of iterations we can perform. This fact is probably due to the critique exposed in Section 2.4, taken from [26]: MMAS spends a lot of computational time ensuring that the pheromone lies in the interval $[\tau_{min}, \tau_{max}]$. This is especially true if the size of the problem is big with respect to the computational time we

Table 3.2: *An approximation of the number of iterations done on average in half an hour and the probability of not choosing the bad arc in any iteration. The number of iterations are taken from the experiments we did. The code we used can be found in [2], and it ran on a Linux machine with a Intel Core i5-3330 CPU and 8GB DDR3 RAM. ((*) Observation: in the case of* rl5934 *with* 10000 *ants the time needed to conclude the first and unique iteration is greater than half an hour. However the algorithm concludes the first iteration before checking the end condition. This is also true in the case of* pr2392 *for the last iteration.)*

Graph	$m = 25$ $((p_{bad}^{not})^{iter})$	$m = 250$ $((p_{bad}^{not})^{iter})$
kroA100	45000 (0.000)	37000 (0.000)
rat783	17000 (0.014)	2300 (0.003)
pr2392	1800 (0.947)	200 (0.947)
rl5934	270 (0.997)	30 (0.998)
Graph	$m = 1000$ $((p_{bad}^{not})^{iter})$	$m = 10000$ $((p_{bad}^{not})^{iter})$
kroA100	25000 (0.000)	2800 (0.000)
rat783	500 (0.007)	50 (0.007)
pr2392	50 (0.948)	5 (0.949) (*)
rl5934	10 (0.998)	1 (0.998) (*)

dispose. Remember that a great number of tours in few iterations emphasizes exploration.

In conclusion of this chapter, we can say that for all the parameters we studied, an additional adaptation for every problem is required. Furthermore, we are convinced that the τ_{max} and τ_{min} parameters should be chosen according to the number of ants, α and β, in order to prevent the experimental bad behaviour described in Section 3.1.6 when we are working

Table 3.3: *An approximation of the number of tours done on average in half an hour with* 25 *and* 10000 *ants and the ratio between the iterations done. The number of iterations are taken from the experiments we did. The code we used can be found in [2], and it ran on a Linux machine with a Intel Core i5-3330 CPU and 8GB DDR3 RAM. ((*) Observation: the value of* $iter(25)/iter(10000)$ *in the case of* rl5934 *seems to be in contrast with the other values that grow with respect to the size of the problem. This is due to the observation done in Table 3.2.)*

Graph	tours $m = 25$	tours $m = 10000$	$iter(25)/iter(10000)$
kroA100	1125000	28000000	16.07
rat783	425000	500000	340
pr2392	45000	50000	360 (*)
rl5934	6750	10000	270 (*)

with few ants and low probabilities. Ideally, the unstable regions in Figure 3.7 (right hand picture) should be covered from the unattainable regions in Figure 3.8. What we proposed is to calculate the proportion τ_{max}/τ_{min} basing the reasoning on the probability of not choosing an arc with the minimum pheromone, rather than on the probability of choosing the arc with the maximum pheromone value, as done in [35]. It is important to notice that it is actually possible to calculate τ_{max} and τ_{min} values for a given problem before the start of the algorithm using the formulas described in this chapter. Thus it is possible also to calculate in advance the probability p_{not}^{bad} of not choosing a bad arc. We stress that our proposal is based also on the experimental problem of the double machine-precision, when the τ_{min} parameter can lose its usefulness. This is especially true when we consider big problems: in that case, the original value of τ_{max}/τ_{min} seems to be extremely high, and we are convinced that reducing it can improve the results. In Chapter 4 we will present some tests with differ-

ent problems that have very different sizes, from 100 till 5934. We used different number of ants and we have reduced the proportion τ_{max}/τ_{min}, giving it a different value at each try. More experiments should be done in the future to understand better this proportion in function of the other parameters, nevertheless the results we obtained are promising, confirming our hypothesis.

Chapter 4

Some Experiments

In this section we want to test the conclusions obtained in Chapter 3 and in particular in Section 3.2. We recall that we supposed that the number of ants needs to be studied more carefully and that the τ_{max}/τ_{min} proportion is too high, especially for big problems. In order to demonstrate these conclusions we used four classical problems found at the web site [1]. The problems have different sizes, described in the numbers contained in the name itself, and we have chosen them as they are commonly studied in many articles in literature. Furthermore we used very different graph sizes to better study the intuitions we gave in the previous sections. The four problems are the following:

- kroA100
- rat783
- pr2392
- rl5934

The algorithm we used can be found at the web site [2]. In particular we used version 1.0.1. The code, written by Adrian Wilke, is a Java

version of the original code written in C by Thomas Stützle that can be found at the web site [23]. The 1.0.1 version we used in Java corresponds to version 1.0.3 of ACOTSP in C that can be found in [23]. The code implements several ACO algorithms apart from MMAS and it is dedicated to the Traveling Salesman Problem. A brief description of the code can be found in [11]. The code is written in order to have high performance with respect to the management of the memory, in particular the pheromone matrix, as described in the web site [23]:

[...] Aim of the software: Provide an implementation of ACO algorithms for the symmetric TSP under one common framework. The implementation is reasonably high performing.

The code is commonly used in the literature, for example in [26, 5, 22, 8, 27, 28] and others. Here we summarize the changes we made in the code, making it more similar to the theory we have analysed:

- new statistical logging is performed at each iteration (the reduction of performance can be observed just for the kroA100 problem, that did the most iterations; nevertheless the log was written in all the runs, assuring the correctness of the results we will talk about)

- we removed the feature which allows the next arc to be selected only from a list of the r best arcs at each step; instead, we preferred to give a chance to every arc; this reduces the performance, especially in big problems where the computational time is an important factor, but it agrees with the theory

- we did not use the restart option; this option makes the pheromone matrix reset if the algorithm stagnates

- we used only the iteration best ant or the best so far ant, while the original code combined them

- we did not use local search algorithms
- sometimes, we changed the τ_{min} value, giving an upper bound to the proportion τ_{max}/τ_{min}, as explained in section 3.2

All the tests were done using version 1.6 of Oracle's Java Virtual Machine on a Linux machine with Ubuntu 12.04, 64 bits, 8GB DDR3 RAM and a Intel Core i5-3330 CPU. The standard options of the virtual machine were used, as set by Eclipse Mars release 4.5.0. In particular, one processor with 1GB of RAM was used. With this setting, we were able to use less than 7000 nodes as problem size without having an out-of-memory error, due to the size of the pheromone matrix. Thus, it is possible to run with our configuration rl5934 without problems. Other background processes of Java, such as the garbage collector, do not influence the performance, since they run on free processors.

For each experiment, the algorithm attempted to solve each problem 10 times with a time limit of 1800 seconds (half an hour) for each attempt. All the tests we ran used the following parameters:

- $\alpha = 1$: as explained in Section 2.2.3 and studied better in 3.1, the value $\alpha = 1$ has good theoretical properties (total convergence to the best arc) and it is the value commonly used in the literature.
- $\beta = 2$: as we discussed in Section 2.2.4, the best value for β is problem dependent. Furthermore it is not possible to discover it without a deep and manual study of the problem instance. We used $\beta = 2$ as it is the value most used in the literature.
- $\rho = 0.2$: as discussed in Section 2.2.5, common values are $\rho = 0.02$ if we have enough available time and $\rho = 0.2$ with shorter runtime. In our problems, the kroA100 problem is the only one where the time limit of half an hour is enough to think about using $\rho = 0.02$,

but we preferred to use the same parameters for all the problems. However, as explained in Section 3.1, the influence of this parameter and the influence of τ_{max}/τ_{min} proportion, α and β are in some sense independent. We recall also that the ρ value should be set along with the number of ants, since both depend on the available time and manage the trade-off exploitation/exploration. In the following we will manage this last parameter, maintaining ρ fixed.

4.1 Experiments over the numbers of ants

The first results we present in Figure 4.1 and in Table 4.1, show the number of ants used in our experiments. As observed in the literature (see for example [35]) and shown in these graphs, excluding few exceptions, the "iteration best" option performs much better than the "best so far" one. Thus in the following, we concentrate our attention on the "iteration best" results (the blue bars in Figure 4.1). In Section 2.2.2 the importance of the number of ants with respect to the trade-off exploitation/exploration is explained (for more details see [27]). The relation can be seen inside the formula $iteration \approx tours/m$ where the number of tours is fixed given the available runtime. In order to show the consequences of this formula, we ran all the problems with a different numbers of ants, and as we have mentioned, the best number of ants can vary a lot with respect to the problem size and the time available. Indeed when we have enough computational time, as in the case of kroA100, many more ants can be used, as the best results reached with 10000 ants demonstrates. In pr2392 and in rl5934, 5 or 10 ants are recommended, while in rat783 (a medium problem) 250 ants gives the best result. For what we have observed, best results are achieved when around $2000 \sim 2500$ iterations are done (see also Table 3.2). Indeed, with kroA100 we can reach 2800 iterations even with 10000 ants, while with rl5934 we never reach this number of iterations so 5 or less ants are recommended. In pr2392, the best result is reached with 10

ants performing around 3500 iterations, slightly worse results with 5 ants (4700 iterations), much worse with 25, where we can perform only 1800 iterations, and significantly worse results were achieved with more ants. In rat783, a medium problem, we can see that the best result is reached with 250 ants, when we did 2300 iterations. Furthermore, it is important to remember the critique in [26]: since the MMAS spends a lot of time managing the pheromone matrix, the iterations done are not proportional with the number of ants. As shown in Tables 3.2 and 3.3, this result is confirmed observing for example, that in rl5934 we reached only 650 iterations with 5 ants, and 520 with 10. So in some extreme cases, the increase of iterations could not compensate the reduction of number of ants.

Finally, with $\alpha = 1$, $\beta = 2$, $\rho = 0.2$, τ_{max}/τ_{min} as defined in equation 3.3 and 1800 seconds of available time, the following ants are recommended: 10000 for kroA100, 250 for rat783, 10 for pr2392 and 5 for rl5934. Thus the often used value of 25 ants can be far away from the best one. As in this context, the best number of iterations seems to be constant in all the tries we did with different problems, it is likely to be the case in other contexts. This new parameter could be a good reference to evaluate the best number of ants to be used. Thus, in order to find the best choice we would recommend first calculating the best number of iterations using a set of little problems, and then choosing the number of ants to use in the final big try. We also stress that $2000 \sim 2500$ iterations can vary in respect to the parameters used (probably mainly with the value of ρ). Anyway, the tests we did are too few to demonstrate this idea, and a deeper study confirming this topic is suggested.

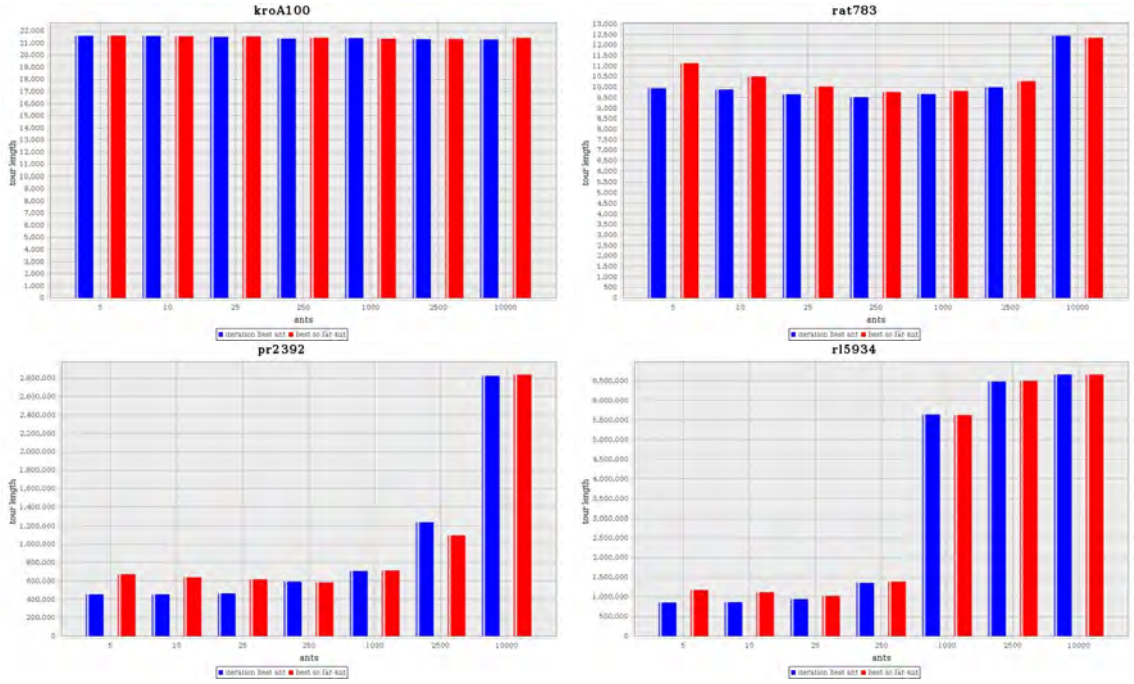

Figure 4.1: *The means of the results of* 10 *runs for* kroA100, rat783, pr2392 *and* rl5934, *with* 5, 10, 25, 250, 1000, 2500 *and* 10000 *ants. Blue bars represent tries with the "iteration best" option and red bars with the "best so far" option. The parameters are* $\alpha = 1$, $\beta = 2$, $\rho = 0.2$ *and* τ_{max}/τ_{min} *as in [35]. All the runs ended after half an hour.*

4.2 Experiments over the proportion τ_{max}/τ_{min}

The second experiment we did regards the τ_{max}/τ_{min} proportion: we wanted to test the arguments discussed in Section 3.2. In particular, we wanted to confirm our proposal that the proportion in Equation 3.3 and hence the probabilities described in Equations 3.4 and 3.5 are too high, and we suggested that reducing them can improve the results. We want also to study better the observations done in Section 3.1.6. We observed that a small number of ants creates "unstable regions" where the unstable fixed-points

Table 4.1: *The general results of* 10 *runs done with different numbers of ants. The parameters are* $\alpha = 1$, $\beta = 2$, $\rho = 0.2$ *and* τ_{max}/τ_{min} *as in [35]. All the runs ended after half an hour. We give here just the results for the "iteration best" option. The values represent the mean tour length of the* 10 *results, while the values in brackets represent the variance.*

Graph	$m = 5$	$m = 10$	$m = 25$
kroA100	21603.7 (1.38)	21588.0 (1.43)	21517.0 (1.72)
rat783	9943.6 (1.39)	9885.9 (0.94)	9660.6 (0.94)
pr2392	454248.3 (1.47)	454084.7 (1.33)	464938.3 (1.06)
rl5934	855519.5 (1.72)	866055.9 (2.22)	945724.7 (3.37)

Graph	$m = 250$	$m = 1000$
kroA100	21372.3 (0.68)	21413.4 (0.71)
rat783	9520.6 (0.87)	9671.0 (0.70)
pr2392	593115.7 (3.01)	708933.3 (1.40)
rl5934	1357382.3 (1.49)	5643464.3 (0.72)

Graph	$m = 2500$	$m = 10000$
kroA100	21316.7 (0.20)	21296.2 (0.14)
rat783	9990.7 (1.33)	12452.8 (1.17)
pr2392	1239696.9 (1.44)	2826468.6 (0.65)
rl5934	6484528.2 (0.75)	6666070.7 (0.64)

are reached. Here, as shown in Figure 3.7 (right hand picture) we noticed an inverted behaviour in the convergence in the top and in the bottom of the picture. This bad behaviour should be moderated by the τ_{max} and τ_{min} parameters that create unattainable regions that exclude the unstable ones (see black regions in Figure 3.8). Thus, we have guessed in Section 3.2 that the black regions are too small, and we have tested what happens when increasing them (that means a reduction of τ_{max}/τ_{min}). The problem can

derive from two causes: the first one is theoretical, that bigger τ_{max}/τ_{min} proportion needs a longer available time as well as bigger number of ants; the second one is experimental, that in big problems the proportion (together with the heuristic function) can create such small probabilities that in a double precision machine they are equivalent to null. Remember that this fact nullifies the effect of τ_{min}. Hence, to demonstrate these conclusions we hope to see the following results:

H1 the reduction of the τ_{max}/τ_{min} proportion should be more effective when few ants are used

H2 the reduction of the τ_{max}/τ_{min} proportion should be more effective when the original one in Formula 3.3 is high in respect to the double machine-precision (we recall that this happens when big problems are considered)

We executed the experiments with the same parameters as described before ($\alpha = 1, \beta = 2, \rho = 0.2$), setting an upper bound to τ_{max}/τ_{min}. We recall that the theoretical Formulas 2.2 and 2.3 are dynamically calculated, as explained in Section 2.4. Thus, we have used an "upper bound" rather than a "fixed value" in order to reduce the proportion τ_{max}/τ_{min}. In particular we used the following bounds:

- kroA100 : 1000
- rat783 : 1000 and 10000
- pr2392 : 1000, 10000 and 100000
- rl5934 : 1000, 10000, 100000 and 1000000

We will first discuss the four problems separately, then we will try to give some general advice.

Starting with kroA100, we can see in Figure 4.2 on left hand graphs and in Table 4.2 that we did not really improve the algorithm, and the results are statistically equivalent (or even just a little bit worse), even if the general trend of the algorithm has not changed. The explication of this is that, since this problem is small in respect to the iterations we did, it does not really matter the configuration of other parameters. As we have just noticed, in this case the only way to really improve the result is by increasing the number of ants. Furthermore, according to H2, the original value of τ_{max}/τ_{min} is quite small in this case, and it does not imply any rounding problem in a double precision machine, as already shown in Table 3.1. Even if we use a shorter runtime, the results are similar. We simulated that case stopping the algorithm at the 2000th iteration, as can be seen in Figure 4.2 on right hand graphs.

Table 4.2: *The general results of the tests done for the problem* kroA100. *The parameters are* $\alpha = 1$, $\beta = 2$ *and* $\rho = 0.2$. *All the tries ended after half an hour. Only the "iteration best" option is considered. The values represent the means of the* 10 *results, while the values in brackets represent the variance.*

N. Ants	10	25	250
original ($\approx$ 1660)	21588.0 (1.43)	21517.0 (1.72)	21372.3 (0.68)
max prop. 1000	21589.2 (1.57)	21555.4 (1.37)	21390.3 (0.69)

In rat783, which we think is a more relevant problem since it is of medium size according to our available time, the situation is much clearer. The 1000 upper bound is too low, and it gives worse results, while the 10000 option gives statistically better results in all the tries we did, as is shown in Table 4.3 and in Figure 4.2 (see left hand side graphs). The mean results decreased as well as the variance. Also in this case, the dif-

ferent τ_{max}/τ_{min} proportion does not change the general trend. According to what we said and to Formula 3.4, the importance of the reduction of τ_{max}/τ_{min} increases when the number of ants is reduced, although also with 250 ants a quite good improvement is still evident. This is in agreement with H1. Since the results regarding the number of ants do not seem to change, we confirm 250 as the best choice. Thus in this case, the 10000 upper bound is the best option. In relation to the original proportion, reducing it by a factor of $\approx$ 10 allows gaining respectively 5.42% (10 ants), 3.77% (25 ants) and 2.54% (250 ants) in respect to the mean results. As a last note, we achieved the same results if the time available decreases. This case is simulated in Figure 4.3, in right hand graphs, where we stopped the algorithm after 2000 iterations in the two top-right graphs, and the last one on the bottom-right after 200. Here, the reduction option continues giving the best results, especially when working with a small number of ants.

Table 4.3: *The general results of the tests done for the problem* rat783. *The parameters are* $\alpha = 1$, $\beta = 2$ *and* $\rho = 0.2$. *All the tries ended after half an hour. Only the "iteration best" option is considered. The values represent the means of the* 10 *results, while the values in brackets represent the variance.*

N. Ants	10	25	250
original ($\approx$ 102261)	9885.9 (0.94)	9660.6 (0.94)	9520.6 (0.87)
max prop. 1000	10510.8 (1.53)	9618.7 (2.76)	9601.5 (2.03)
max prop. 10000	9349.3 (0.90)	9296.0 (0.50)	9278.8 (0.45)

The third problem we studied is pr2392, as described in Table 4.4 and in Figure 4.4 (left hand graphs). First, notice that, confirming H1, the importance of our reduction of τ_{max}/τ_{min} decreases with the increase in

the number of ants, and all the results tend to reach the same best result. The only exception is the 1000 upper bound, that drastically reduces its result with 250 ants. This is due to the fact that, since in this case we can perform really few iterations (see Table 3.2), we should concentrate on the exploitation of the first found solutions, even if they are not the best ones. This is in agreement with the observation made in Section 3.1, in particular those related with Figure 3.4: the local optimum are not casually disposed, then the exploitation is more important than exploration, if we have a small runtime. Hence we must exclude quickly the arcs not chosen, giving them a small τ_{min}. With the upper bound of 10000 the situation is still worse, but we can observe that it tends to be in agreement with the other tries if the number of ants increases (again, it agrees with H1). Finally, considering the case of 100000, we reached better mean results in every situation. Concerning the variance, it is slightly worse with 10 ants (that we recall is the best choice of number of ants in our tests), almost equal with 25 ants, and much better with 250. The little growth of the variance using 10 ants can be explained as, giving a bigger τ_{min}, we explore more different tours, but the available time is not enough to exploit all the found solutions. Nevertheless, the difference we observed in the variance of $+0.32\%$ is not statistically relevant, as it is calculated in just 10 tries. Finally, also in this test, reducing the original proportion by a factor of $\approx$ 10 is recommended, since we can gain 1.09% (10 ants), 1.94% (25 ants) and 0.44% (250 ants) in respect to the mean results. In Figure 4.4 in right graphs, where we have interrupted the execution of the algorithm at iteration 200, and at iteration 50 for the bottom right one, our proportion reduction seems to continue giving best results. Again, the changes in the proportion do not change the general trend of the algorithm given a shorter runtime.

The last problem we studied is rl5934. Observing Figure 4.5 (left hand graphs) and Table 4.5, we can repeat some observations we did for the previous problems and according to H1. The general trend of the algo-

Table 4.4: *The general results of the tests done for the problem* pr2392. *The parameters are* $\alpha = 1$, $\beta = 2$ *and* $\rho = 0.2$. *All the runs ended after half an hour. Only the "iteration best" option is considered. The values represent the means of the* 10 *results, while the values in brackets represent the variance.*

N. Ants	10	25	250
original ($\approx$ 954770)	454084.7 (1.33)	464938.3 (1.06)	593115.7 (3.01)
max prop. 1000	649538.0 (1.48)	639687.3 (1.58)	687806.5 (1.22)
max prop. 10000	493296.4 (2.29)	488813.0 (4.13)	599074.1 (2.32)
max prop. 100000	449135.7 (1.65)	455934.3 (1.03)	590498.0 (2.02)

rithm remains equal to the original one and the upper bound of 1000 is too low in this example where we can perform few iterations, and its behaviour becomes much worse when we consider 250 ants (that is, just 30 iterations, see Table 3.2). The upper bound of 10000 is too low as well, even if its behaviour tends to adapt to the other ones when we have more ants, as supposed in H1. The 1000000 case, where we reduced the original proportion by a factor of ≈ 5.87, gives better results in terms of mean and above all in terms of variance: an improvement of 0.55% with 10 ants and 0.80% with 25 ants. In the run with 250 ants, the reduction worsens the result, and we again argue this is due to the low number of iterations done in this case. Lastly, the upper bound 100000 is surprising because, as far as we have seen till now, a reduction by a factor ≈ 10 should give the best result, while in this case we adopted a reduction by a factor of ≈ 58.7 and we have gained respectively 0.79% (10 ants) and 1.27% (25 ants). The case using 250 ants is still worse, as we expected. This experiment is in accordance with H2 that in big problems, the τ_{max}/τ_{min} proportion is too high and should be reduced more than in little problems. The variance

of the last upper bound is not as good as the 1000000 one, but it is still much better in respect to the original one. A deeper study of this case should be done, but our initial guess is that the good results achieved with 100000 as upper bound compared with 1000000, can be due to the double machine-precision: a proportion of 1000000 makes more arcs out of reach. This hypothesis is reinforced by the smaller variance (see Tables 3.1 and 3.2 that illustrate indeed how difficult choosing an arc with τ_{min} can be). As a rule of thumb we can say that also in this example, a reduction by a factor of $\approx$ 10 is suggested, leaving open the possibility of higher increasing of it, according to H2, in respect to the machine precision. Finally, we can see that in Figure 4.5, in the two top-right graphs, where we stopped the algorithm at iteration 50, also in the case of having less available time, the reduction improves the results.

Table 4.5: *The general results of the tests done for the problem* rl5934. *The parameters are* $\alpha = 1$, $\beta = 2$ *and* $\rho = 0.2$. *All the runs ended after half an hour. Only the "iteration best" option is considered. The values represent the means of the* 10 *results, while the values in brackets represent the variance.*

N. Ants	10	25	250
original($\approx$ 5876593)	866055.9(2.22)	945724.7(3.37)	1357382.3(1.49)
max prop.1000	1104303.1(1.80)	1131054.1(2.30)	1582456.5(1.05)
max prop.10000	909019.6(2.24)	951457.4(2.00)	1406002.3(0.94)
max prop.100000	859207.9(1.75)	933725.3(2.33)	1407062.8(1.08)
max prop.1000000	861313.1(0.89)	938203.6(2.08)	1390898.2(1.33)

Summarizing the observations we have analysed for each problem tested, we can say that in general the original proportion in Formula 3.3 is too

high, and reducing it gives better results in terms of both mean and variance. Nevertheless we must be aware of the following extreme cases:

- if the problem size is small, the proportion can be left as in Formula 3.3: it is indeed quite small as well
- if the time available is too short with respect to the number of ants and a good number of iterations can not be achieved, the original proportion seems to give the best results, although it is important to notice the bigger variance obtained

Apart from the explained cases, when we consider runs with a *good* equilibrium between the number of ants and iterations, the best results in mean and variance are obtained with the following proportion (modifying the one in [35]):

$$\tau_{max} = \frac{1}{\rho f(l_b)} \;\; ; \;\; \tau_{min} = \frac{\kappa \cdot \tau_{max}(1 - \sqrt[n]{0.05})}{(\frac{n}{2} - 1)\sqrt[n]{0.05}} \;\; ; \;\; \frac{\tau_{max}}{\tau_{min}} = \frac{(\frac{n}{2} - 1)\sqrt[n]{0.05}}{\kappa \cdot (1 - \sqrt[n]{0.05})} \tag{4.1}$$

From the above experiments, we recommend using $\kappa \approx 10$ as a rule of thumb. The reduction of this proportion is motivated by the basic idea of Max-Min Ant System that we want to emphasize: each arc, even the never selected ones, must have a minimum probability of being chosen bigger than 0. Thus, our reasoning, explained in detail in Section 3.2, takes in consideration the arcs with the minimum value of pheromone, rather than the arcs with the maximum value. In the same section we showed some probabilities that come out in real problems, pointing out that they are so small that are considered 0 in real implementations. This is in contrast with the principle of Max-Min Ant System. Furthermore the considerations in Section 3.2 shows how to relate τ_{max} and τ_{min} with the other parameters α, β and the number of ants. We recall that this relation was not present in the original discussion in [35], that considers

only the size of the problem. Both hypothesis, H1 and H2, that we wanted to demonstrate, seem to be true, even if some more tests are needed to better understand the behaviour of the algorithm in relation to κ. The comparison between experiments kroA100 and rl5934 shows that the best value of κ grows jointly with the size of the problem. We suppose this is due to machine-precision problems, but more experiments are needed, using more precision, to see the resulting effect. Furthermore, a more detailed study is needed to understand the function of growth of κ, and if there are some lower or upper bounds. Regarding the relation with the best number of ants to be used, the value of κ we have suggested seems not to change its value.

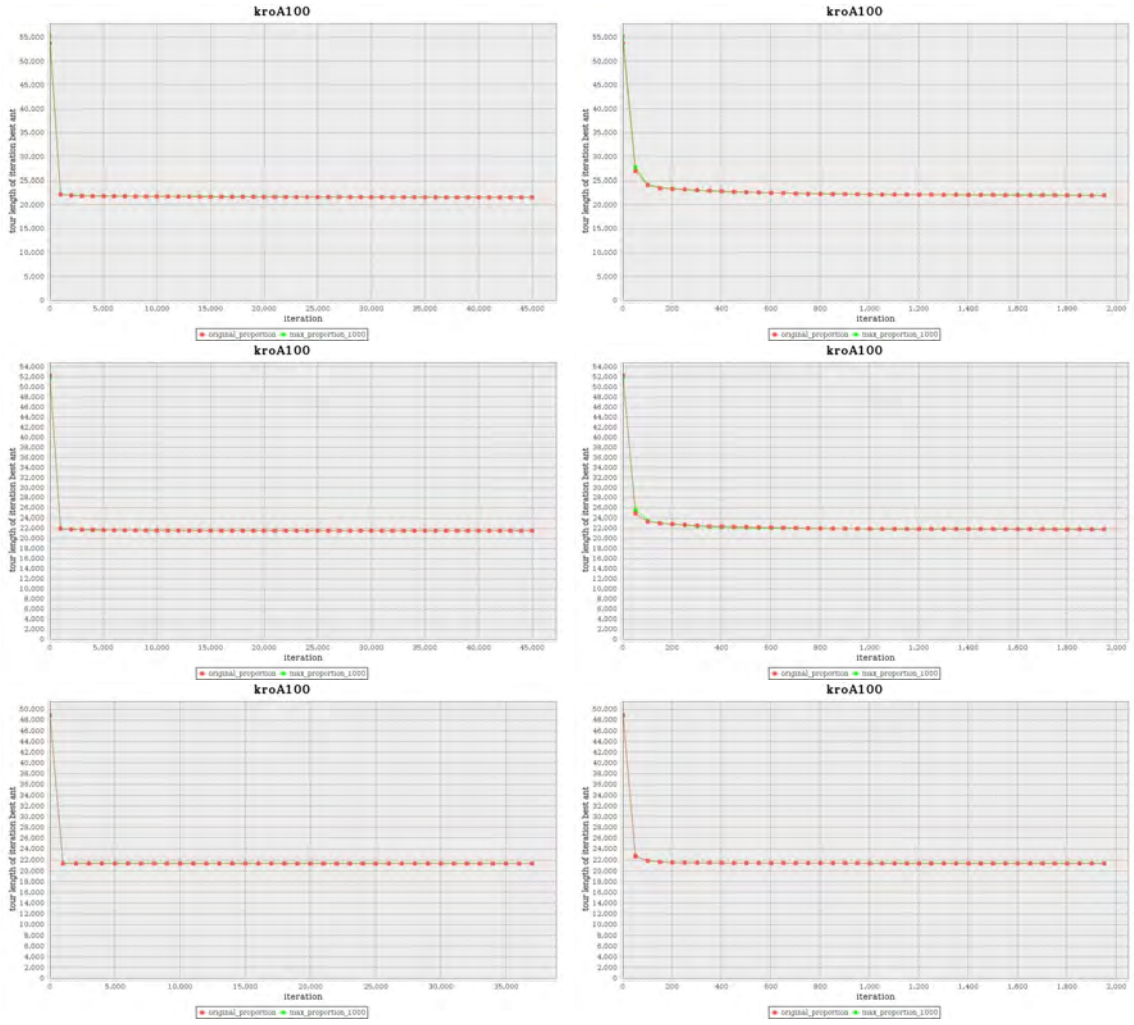

Figure 4.2: *The means of the results of* 10 *runs for the problem* kroA100 *with the original proportion* τ_{max}/τ_{min} *and with the upper bound* 1000, *with* 10 *(graphs on first line),* 25 *(on second line), and* 250 *(on third line) ants, using the "iteration best" option. The parameters are* $\alpha = 1$, $\beta = 2$ *and* $\rho = 0.2$. *The runs on the left ended after half an hour, while the ones on the right were stopped after* 2000 *iterations. The general trend does not change when varying the proportion* τ_{max}/τ_{min}.

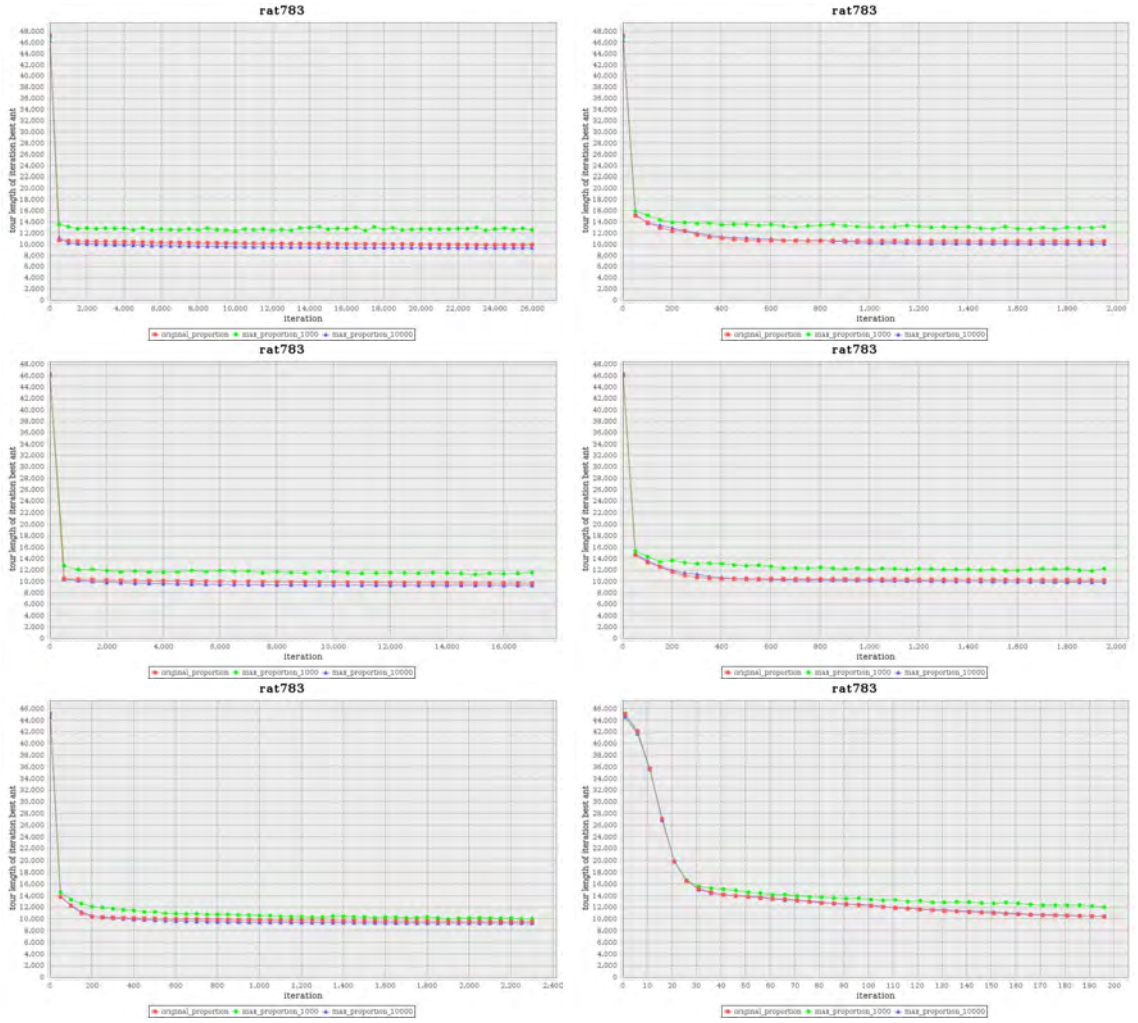

Figure 4.3: *The means of the results of* 10 *runs for the problem* rat783 *with the original proportion* τ_{max}/τ_{min} *and with the upper bounds* 1000 *and* 10000, *with* 10 *(graphs on first line),* 25 *(on second line), and* 250 *(on third line) ants, using the "iteration best" option. The parameters are* $\alpha = 1$, $\beta = 2$ *and* $\rho = 0.2$. *The runs on the left ended after half an hour, while the first two ones on the right were stopped after* 2000 *iterations, and the third one on the right was stopped at* 200 *iterations. The general trend does not change when varying the proportion* τ_{max}/τ_{min}.

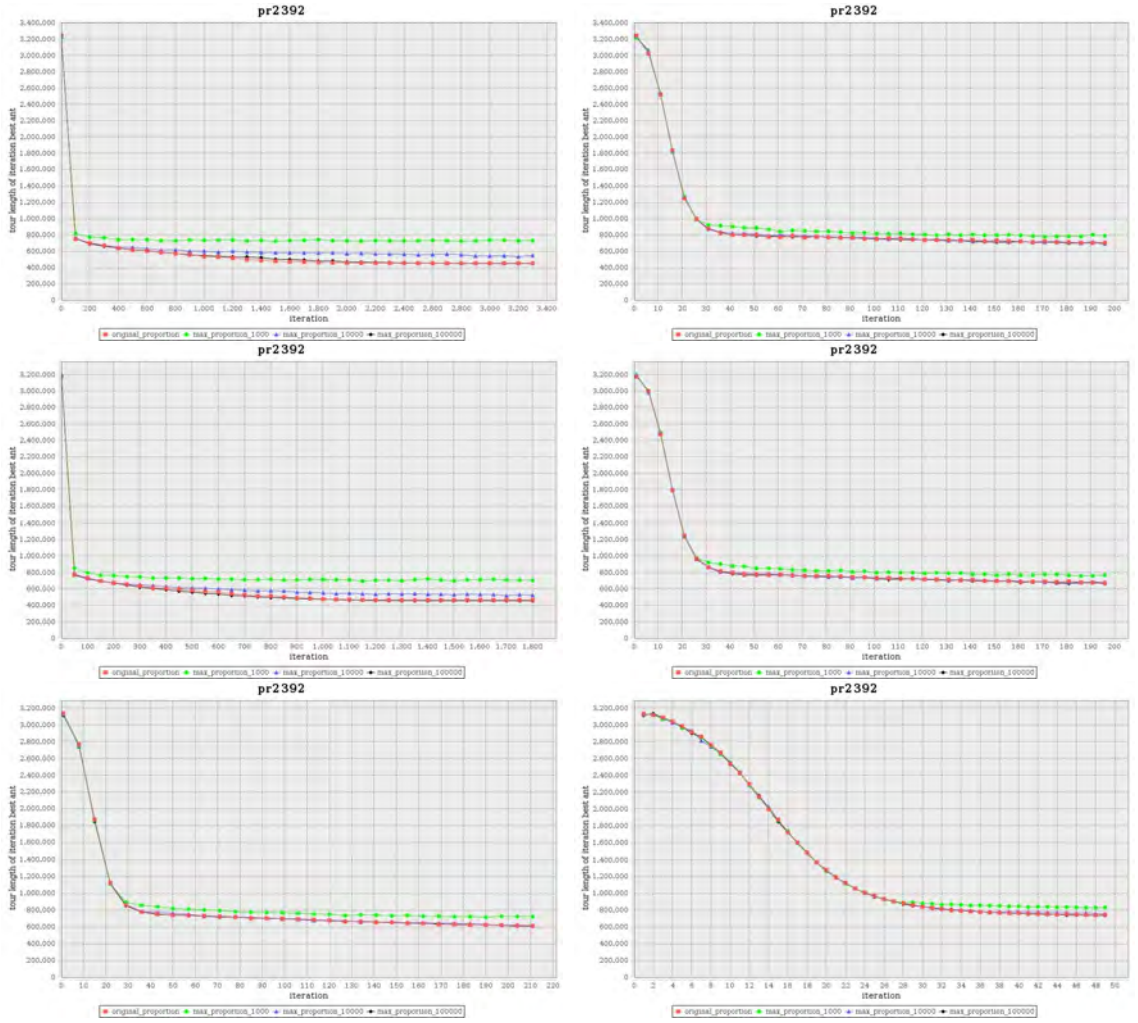

Figure 4.4: *The means of the results of* 10 *runs for the problem* pr2392 *with the original proportion* τ_{max}/τ_{min} *and with the upper bounds* 1000, 10000 *and* 100000, *with* 10 *(graphs on first line),* 25 *(on second line), and* 250 *(on third line) ants, using the "iteration best" option. The parameters are* $\alpha = 1$, $\beta = 2$ *and* $\rho = 0.2$. *The runs on the left ended after half an hour, while the first two ones on the right were stopped after* 200 *iterations and the third one on the right was stopped after* 50 *iterations. The general trend does not change when varying the proportion* τ_{max}/τ_{min}.

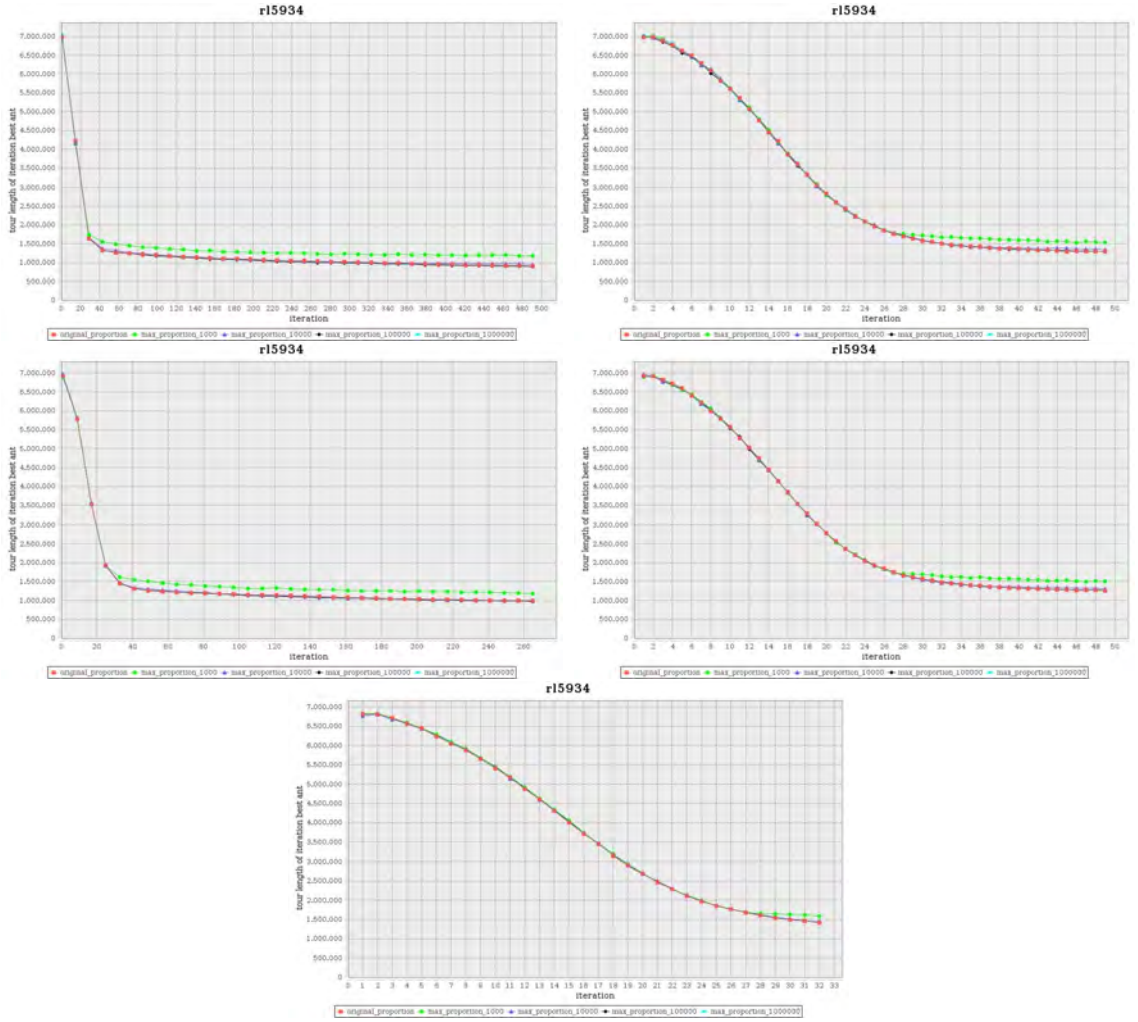

Figure 4.5: *The means of* 10 *runs for the problem* rl5934 *with the original proportion* τ_{max}/τ_{min} *and with the upper bounds* 1000, 10000, 100000 *and* 1000000, *with* 10 *(graphs on first line),* 25 *(on second line), and* 250 *(on third line) ants, using the "iteration best" option. The parameters are* $\alpha = 1$, $\beta = 2$ *and* $\rho = 0.2$. *In the first two lines, the two runs on the left ended after half an hour, while the ones on the right were stopped after* 50 *iterations. The bottom run ended after half an hour. The general trend does not change when varying the proportion* τ_{max}/τ_{min}.

Chapter 5

Conclusions and Future Work

In this thesis we analysed the Max-Min Ant System algorithm when used to solve the Symmetric Euclidean Travelling Salesman Problem. Apart from other articles in the literature, our main results are based on the theory found in [35] and in [24]. The first one describes the theory behind the commonly used values of τ_{max} and τ_{min}, while the second analyses the α parameter from a theoretical point of view, using the stability analysis of a differential equation.

It is well known that the core problem in using ACO algorithms is the setting of parameters. It should be highlighted that no absolute best combination of parameters have been found till now for the MMAS algorithm. We are also convinced that it will not occur in the future, since the parameters are strongly problem dependent. Hence, the capability of humans to understand the problem instance before the running of the algorithm is mandatory. Our work is intended to give some further understanding of this problem.

In the literature there are a lot of studies, theoretical and experimental, about the parameters of the MMAS algorithm, nevertheless they seem to be considered in a standalone way. We tried to study them jointly and we succeed in giving some more relations that have not been previously shown.

First of all we continued the work in [24] over α, generalizing it and analysing all the other parameters. We studied the behaviour of the algorithm when a value of α greater than 1 is used. This is a choice that seems not to be studied in the literature. Our analysis concerned also τ_{max} and τ_{min}, giving a better understanding of their effect from a theoretical point of view. Furthermore, we discussed some differences between theoretical and real situations. In particular we showed how in real situations the number of ants can change the theoretical behaviour in the case that the probability of choosing an arc is near the extremes 0 and 1, rendering it unstable and giving the possibility of falling into local optimum. We analysed the possibility to correct this problem using τ_{max} and τ_{min} parameters.

Then, keeping in mind the number of ants problem, we reflected on the τ_{max}/τ_{min} proportion described in [35]. We noticed that the authors used a formula including only the size of the problem. Instead we wanted to study the problem giving a result that includes all the parameters of MMAS. We illustrated how the original proportion can become useless in real problems. The motivation is that the original formula took in consideration the probability of choosing the arcs with the maximum pheromone value. According to the basic idea of MMAS (each arc must have a minimum probability of being chosen), we focused our attentions on the arcs that have the minimum value of pheromone. The resulting formula can be calculated theoretically, letting the user choose the correct values before the start of the algorithm. Once the relation among all the parameters was established, we studied in more detail a simplification of it, relating only τ_{max}, τ_{min} and the number of ants. It came out that in reality the number of ants plays an important role in the setting of τ_{max} and τ_{min} values.

After that, some experiments were shown, where we studied the correct number of ants given the available computational time and the size of the problem. The experiments showed that the best number of ants can be very different from the commonly used value of 25. In the literature, the studies over the number of ants seem to be qualitative and they do not give a fixed reference point to choose this parameters. Nevertheless in our experiments an interesting relation came out: it seems that the best number of iterations is quite constant in all the tries we did with different problems. If this idea can be confirmed in the future, it could be a good reference point in order to set the number of ants given the size of the problem and the available time.

A part from that, we analysed the effect of our study on the proportion τ_{max}/τ_{min}. Confirming our theoretical study, all of our experiments suggested that reducing the proportion given in [35] by a (rule of thumb) factor of 10 gives better results, in terms of mean and variance of the tries done. Moreover, the experiments suggest that the factor of 10 can be even higher according to the size of the problem, confirming our hypothesis. Apart from the theoretical considerations, we analysed also the effect of the double machine-precision problem. Even though the best reduction factor should be studied in more detail, it appears to be clear that the original proportion in most cases is too high.

Finally, we leave open the following questions that we hope to study in the future:

- even if $\alpha \leq 1$ is commonly accepted, the case of $\alpha \gg 1$ shows some asymptotes that could be interesting to study; we think that, in this case, it is possible to achieve "good" solutions in situations with a *really short* computational time, calculating a priori the probability to be trapped in a bad solution

- using the best number of iterations as a reference point, the best number of ants can be calculated; we observed this possibility in our few experiments, but more tests are needed in order to accept or reject this hypothesis

- the influence of the new parameter κ in the proportion τ_{max}/τ_{min} must be studied to reach a more precise value; furthermore, we suggest analysing its behaviour when the size of the problem grows, including rounding problems in a double precision machine.

Bibliography

[1] A repository of instances for the TSP of various types from the University of Heidelberg , *http://www.iwr.uni-heidelberg.de/groups/comopt/software/tsplib95/tsp/*.

[2] ACOTSPJava, *http://adibaba.github.io/acotspjava/*.

[3] Mohammed Alhanjouri and Belal Alfarra, *Ant colony versus genetic algorithm based on travelling salesman problem*, International Journal of Computer Technology and Applications **2**, no. 3, 570–578.

[4] Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp, *A racing algorithm for configuring metaheuristics.*, GECCO, Morgan Kaufmann, 2002, pp. 11–18.

[5] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle, *F-race and iterated f-race: An overview*, IRIDIA – Technical Report Series **Technical Report No. TR/IRIDIA/2009-018** (2009).

[6] Christian Blum, *Ant colony optimization: Introduction and recent trends*, Physics of Life Reviews **2** (2005), 353–373.

[7] Marco Dorigo, Mauro Birattari, and Thomas Stützle, *Ant colony optimization - artificial ants as a computational intelligence technique*, IRIDIA – Technical Report Series **Technical Report No. TR/IRIDIA/2006-023** (2006).

[8] Marco Dorigo and Christian Blum, *Ant colony optimization theory: A survey*, Theoretical Computer Science **344** (2005), 243–278.

[9] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni, *Ant system: Optimization by a colony of cooperating agents*, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B CYBERNETICS **25** (1996), no. 1.

[10] Marco Dorigo and Thomas Stützle, *The ant colony optimization metaheuristic: Algorithms, applications, and advances*, Handbook of Metaheuristics, Kluwer Academic Press, 2003, pp. 251–286.

[11] Marco Dorigo and Thomas Stützle, *Ant colony optimization*, Bradford Company, Scituate, MA, USA, 2004.

[12] Dorian Gaertner and Keith L. Clark, *On optimal parameters for ant colony optimization algorithms.*, IC-AI, CSREA Press, 2005, pp. 83–89.

[13] S. Goss, S. Aron, J.L. Deneubourg, and J.M. Pasteels, *Self-organized shortcuts in the argentine ant*, Naturwissenschaften **76** (1989), no. 12, 579–581.

[14] Walter J. Gutjhar, *A generalized convergence result for the graph–based ant system metaheuristic.*

[15] ———, *Mathematical runtime analysis of ACO algorithms: survey on an emerging issue*, Swarm Intelligence **1** (2007), no. 1, 59–79.

[16] ———, *First steps to the runtime complexity analysis of ant colony optimization*, Computers & OR **35** (2008), no. 9, 2711–2727.

[17] Zhi-Feng Hao, Rui-Chu Cai, and Han Huang, *An adaptive parameter control strategy for aco*, Fifth International Conference on Machine Learning and Cybernetics (2006).

[18] S. Kirkpatrick and G. Toulouse, *Configuration space analysis of travelling salesman problems*, (1985).

[19] Kuan Yew Wong Komarudin, *Parameter tuning for ant colony optimization: A review*, International Conference on Computer and Communication Engineering (2008).

[20] Timo Kötzing, Frank Neumann, Heiko Röglin, and Carsten Witt, *Theoretical properties of two aco approaches for the traveling salesman problem.*, ANTS Conference, Lecture Notes in Computer Science, vol. 6234, Springer, 2010, pp. 324–335.

[21] Bifan Li, Lipo Wang, and Wu Song, *Ant colony optimization for the traveling salesman problem based on ants with memory*, Fourth International Conference on Natural Computation (2008), 496–501.

[22] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Thomas Stützle, and Marco Birattari, *The irace package: Iterated racing for automatic algorithm configuration*, IRIDIA – Technical Report Series **Technical Report No. TR/IRIDIA/2011-004** (2011).

[23] aco metaheuristic, *http://www.aco-metaheuristic.org/aco-code/public-software.html*.

[24] Bernd Meyer, *On the convergence behaviour of ant colony search*, Complexity International **12** (2005).

[25] Alberto Moraglio, Fernando E. B. Otero, and Colin G. Johnson, *The aco encoding*.

[26] Sabrina Oliveira, Mohamed Saifullah Hussin, Thomas Stützle, Andrea Roli, and Marco Dorigo, *A detailed analysis of the population-based ant colony optimization algorithm for the tsp and the qap*, IRIDIA – Technical Report Series **Technical Report No. TR/IRIDIA/2011-006** (2011).

[27] Paola Pellegrini, Daniela Favaretto, and Elena Moretti, *On max – min ant system's parameters*, Ant Colony Optimization and Swarm Intelligence, Lecture Notes in Computer Science, vol. 4150, Springer Berlin Heidelberg, 2006, pp. 203–214.

[28] Paola Pellegrini, Thomas Stützle, and Mauro Birattari, *Off-line vs. on-line tuning: A study on max–min ant system for the tsp.*, ANTS Conference, Lecture Notes in Computer Science, vol. 6234, Springer, 2010, pp. 239–250.

[29] ———, *A critical analysis of parameter adaptation in ant colony optimization.*, Swarm Intelligence **6** (2012), no. 1, 23–48.

[30] Marcin L. Pilat and Tony White, *Using genetic algorithms to optimize acs-tsp.*, Ant Algorithms, Lecture Notes in Computer Science, vol. 2463, Springer, 2002, pp. 282–287.

[31] Marcus Randall, *Near parameter free ant colony optimisation.*, ANTS Workshop, Lecture Notes in Computer Science, vol. 3172, Springer, 2004, pp. 374–381.

[32] Enda Ridge and Daniel Kudenko, *Tuning the performance of the mmas heuristic.*

[33] Thomas Stützle and Marco Dorigo, *Aco algorithms for the traveling salesman problem.*

[34] ———, *A short convergence proof for a class of ant colony optimization algorithms*, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION **6** (2002), no. 4.

[35] Thomas Stützle and Holger H. Hoos, *Max–min ant system*, Future Generation Computer Systems **16** (2000), 889–914.

[36] Thomas Stützle, Manuel López-Ibáñez, Paola Pellegrini, Michael Maur, Marco Montes De Oca, Mauro Birattari, and Marco Dorigo, *Parameter adaptation in ant colony optimization*, IRIDIA – Technical Report Series **Technical Report No. TR/IRIDIA/2010-002** (2010).

[37] Zainudin Zukhri and Irving Vitra Paputungan, *A hybrid optimization algorithm based on genetic algorithm and ant colony optimization*, International Journal of Artificial Intelligence & Applications **4** (2013), no. 5, 63–75.